

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/331310518>

# programming technique lecture

Presentation · February 2019

---

CITATIONS

0

READS

1,895

1 author:



nguyen dinh Cuong

Nha Trang University

41 PUBLICATIONS 18 CITATIONS

SEE PROFILE

# NGÔN NGỮ LẬP TRÌNH C/C++

**Nguyễn Đình Cường**

**Nguyễn Đình Thuân**

**Khoa Công Nghệ Thông Tin**

**Đại học Nha Trang**

Nha Trang, 7-2008

# Nội dung môn học

Chương 1: Tổng quan

Chương 2: Các toán tử

Chương 3: Các cấu trúc điều khiển

Chương 4: Dữ liệu có cấu trúc

Chương 5: Các hàm trong C

Chương 6: Các cấu trúc dữ liệu khác

Chương 7: Đồ họa trong C

# Chương 1: Tổng quan

## 1.1 Giới thiệu

- Đã có nhiều sách trên thế giới viết về C/C++ và hầu hết là ca ngợi, nhất là các lập trình viên luôn xem C/C++ là công cụ mạnh và uyển chuyển.
- C là kết quả của quá trình phát triển khởi đầu từ ngôn ngữ BCPL (do Martin Richards đưa ra vào năm 1967) là sản phẩm của dự án Combine Programming Language giữa 2 trường Đại học London và Cambridge. Ngôn ngữ B (do Ken Thompson phát triển từ ngôn ngữ BCPL vào năm 1970 khi viết hệ điều hành UNIX đầu tiên trên máy PDP-7) và ngôn ngữ B là tiền thân của ngôn ngữ C.

# 1.1 Giới thiệu (tiếp)

- Năm 1978, hai tác giả Brian Kernighan và Dennis Ritchie và đã cho xuất bản quyển The C Programming Language (Prentice-Hall) và được phổ biến rộng rãi đến nay. Vì vậy ngôn ngữ C thường được gán cho “Tiêu chuẩn K&R”.
- Hiện nay có gần 30 trình biên dịch C đang phổ biến trên thị trường và chúng không nhất quán nhau. Để cải thiện tình trạng này, chuẩn ANSI C cũng được ra đời vào năm 1978, nhằm chăm lo việc phát triển các môi trường và các hàm thư viện của C.

# Các đặc điểm của ngôn ngữ C:

- o Tính cô đọng (compact): C chỉ có 32 từ khóa chuẩn và 40 toán tử chuẩn, nhưng hầu hết đều được biểu diễn bằng những chuỗi ký tự ngắn gọn.
- o Tính cấu trúc (structured): C có một tập hợp những chỉ thị của lập trình như cấu trúc lựa chọn, lặp... Từ đó các chương trình viết bằng C được tổ chức rõ ràng, dễ hiểu.
- o Tính tương thích (compatible): C có bộ tiền xử lý và một thư viện chuẩn vô cùng phong phú nên khi chuyển từ máy tính này sang máy tính khác các chương trình viết bằng C vẫn hoàn toàn tương thích.
- o Tính linh động (flexible): C là một ngôn ngữ rất uyển chuyển và cú pháp, chấp nhận nhiều cách thể hiện, có thể thu gọn kích thước của các mã lệnh làm chương trình chạy nhanh hơn.
- o Biên dịch (compile): C cho phép biên dịch nhiều tập tin chương trình riêng rẽ thành các tập tin đối tượng (object) và liên kết (link) các đối tượng đó lại với nhau thành một chương trình có thể thực thi được (executable) thống nhất.

## 1.2 Môi trường làm việc Turbo C

1. Gọi Turbo C
2. Soạn thảo chương trình mới
3. Ghi chương trình đang soạn thảo vào đĩa
4. Thực hiện chương trình
5. Mở một chương trình đã có trên đĩa
6. Thoát khỏi Turbo C và trở về DOS (hay Windows)
7. Sử dụng một số lệnh trên thanh menu

## 1.3 Các thành phần trong chương trình C

- Bộ ký tự
- Các từ khoá trong C
- Lời chú thích đặt trong cặp dấu /\* và \*/  
hoặc sau //

# Ví dụ 1:

```
/*VIDU.CPP*/
#include <stdio.h>

int main()
{
    printf("Day la vi du \n");
    printf("don gian Lap trinh C\n");

    return 0;
}
```

**Ghi chú**

**Thư viện nhập xuất chuẩn**

**Hàm main**

**Báo CT kết thúc cho HĐH**

## Ví dụ 2

Khai báo 2 biến số nguyên, “a” và “b”

Nhập 2 số nguyên vào a và b

Viết các biểu thức “a”, “b” và “a-b” theo định dạng %d

```
#include <stdio.h>
#include <conio.h>

int main(void)
{
    int a, b;
    printf("Nhập 2 số nguyên: ");
    scanf("%d %d", &a, &b);
    printf("%d - %d = %d\n", a, b, a - b);
    getch();
    return 0;
}
```

```
Nhap 2 so nguyen: 21 17
21 - 17 = 4
```

## Ghi chú:

- Phần chú thích được trình biên dịch bỏ qua
- Các từ có phân biệt chữ hoa và chữ thường
- Câu lệnh luôn được kết thúc bằng dấu ;
- Chuỗi ký tự phải ghi giữa cặp nháy kép “ ”
- In xuống dòng dùng ký tự \n
- Chương trình C gồm 1 hoặc nhiều hàm, hàm được gọi thực hiện đầu tiên là hàm main.

## 1.4 Các bước cơ bản khi viết chương trình

1. Phân tích, đặc tả bài toán
2. Tìm lời giải (thuật toán) và kiểu dữ liệu.
3. Viết chương trình bằng ngôn ngữ lập trình
4. Chạy thử sửa lỗi.
5. Tổng kết chương trình

# 1.5 Các kiểu dữ liệu cơ bản trong C

Kiểu	Tùy khóa	Kích thước
Ký tự	char	1 byte
Số nguyên	int	2 bytes
Số thực	float	4 bytes
Số thực chính xác kép	double	8 bytes
Không giá trị	void	

# Kiểu logic trong C

- Trong C không có kiểu dữ liệu logic (nhận các giá trị ĐÚNG – SAI), thay vào đó các biểu thức so sánh sẽ cho kết quả là **SỐ**
- Biểu thức có giá trị **0** (0.0) ứng với kết quả SAI (FALSE)
- Biểu thức có giá trị khác không như : **1, 3.5, -7, 10.4, ...** đều được xem là ĐÚNG (TRUE)

## 1.5 Các kiểu dữ liệu cơ bản (tiếp)

### Bộ chuyển kiểu (modifiers)

**signed** (có dấu)

**unsigned** (không dấu)

**short** (số nguyên ngắn)

**long** (số nguyên độ dài gấp đôi)

# DataType

Type	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4 * (10**-38) to 3.4 * (10**+38)
double	64 bits	1.7 * (10**-308) to 1.7 * (10**+308)
long double	80 bits	3.4 * (10**-4932) to 1.1 * (10**+4932)

## 1.6 Khai báo trong C

- Tất cả các yếu tố trong chương trình do người lập trình đặt ra phải được khai báo trước khi sử dụng, khai báo trước hết phải đặt tên cho yếu tố đó.
- Tên hay còn gọi là danh hiệu(identifier) dùng để đặt cho biến, hằng, kiểu, hàm, ... Tên được đặt theo qui định:
  - Gồm chữ cái, chữ số, dấu gạch chân.
  - Không bắt đầu bằng chữ số
  - Không trùng với từ khóa.
- Tên chuẩn là một số tên do C đặt sẵn như: sin, cos...
- Độ dài tối đa của tên là không giới hạn, tuy nhiên chỉ có 31 ký tự đầu tiên là có ý nghĩa.

# 1. Khai báo biến

- Khai báo `<kiểu> <tên biến>;`
- Biến là đại lượng được người lập trình định nghĩa và được đặt tên thông qua việc khai báo biến. Biến dùng để chứa dữ liệu trong quá trình thực hiện chương trình và giá trị của biến có thể thay đổi trong quá trình này.
- Mỗi biến thuộc về một kiểu dữ liệu xác định và có giá trị thuộc kiểu đó.

# 1. Khai báo biến (tiếp)

Ví dụ:

```
int a, b, c; /*Ba biến a, b,c có kiểu int*/  
long int chu_vi; /*Biến chu_vi có kiểu long*/  
float nua_chu_vi; /*Biến nua_chu_vi có kiểu float*/  
double dien_tich; /*Biến dien_tich có kiểu double*/
```

- a) Khai báo biến ngoài (biến toàn cục): Các biến được đặt bên ngoài tất cả các hàm(kể cả hàm main) và phạm vi sử dụng trong toàn bộ chương trình.
- b) Khai báo biến trong(biến cục bộ): Các biến được đặt ở bên trong hàm hay khối lệnh. Các biến này chỉ có tác dụng trong hàm hoặc khối lệnh tương ứng

# 1. Khai báo biến (tiếp)

Ví dụ 1:

```
#include <stdio.h>
#include<conio.h>
int a; //khai bao bien ngoai
int main ()
{
    int i,j; //khai bao bien ben trong hàm main
    clrscr();
    i=1; j=2;
    a=3;
    printf("\n Gia tri cua i la %d",i);
    printf("\n Gia tri cua j la %d",j);
    printf("\n Gia tri cua bienngoai a la %d",a);
    getch();
    return 0;
}
```

## 2. Khai báo hằng

Khai báo

```
const [kiểu] <tên hằng> = <giá trị>;
```

Hoặc được khai báo thông qua gán giá trị đầu

- Hằng (Constant) là đại lượng không đổi trong quá trình thực thi của chương trình.
- Hằng bao gồm hằng số nguyên, hằng số thực, hằng ký tự, hằng chuỗi ký tự.

a) Hằng số:

Hằng số nguyên: 10, -167

Hằng số thực: 1.234, -0.34E3

- Ngầm định, trình biên dịch ghép hằng vào kiểu dữ liệu tương ứng nhỏ nhất

Ví dụ:      hằng số 10 có kiểu int

                hằng số 60000 có kiểu unsigned

                hằng số 100000 có kiểu long

- C qui ước các hằng số thực có kiểu double

## 2. Khai báo hàng (tiếp)

# Ví dụ về hằng

Các hằng pi, t, heso được tạo với từ khóa const

```
#include <stdio.h>

int main(void)
{
    const long double pi = 3.141592653590L;
    const t = 7;
    const heso = 9.123;

    days_in_week = 5;

    return 0;
}
```

Lỗi

## b) Hằng ký tự

- Hằng ký tự là một ký tự riêng biệt được viết trong cặp dấu nháy đơn (' '). Mỗi một ký tự tương ứng với một giá trị trong bảng mã ASCII. Hằng ký tự cũng được xem như trị số nguyên.

Ví dụ: 'a', 'A', '0', '9'

- Có thể thực hiện các phép toán số học trên 2 ký tự (thực chất là thực hiện phép toán trên giá trị ASCII của chúng)

## b) Hằng ký tự(tiếp)

Một số ký tự không in được (có trị ASCII từ 0 đến 31) trình biên dịch C nhận biết điều này bằng cặp ký tự bắt đầu bằng '\':

\n newline

\t tab

\b backspace

\r carriage return

\f form feed

\a alert

\' '

\\" "

\\\ \

# Hằng chuỗi ký tự

Hằng chuỗi ký tự là một chuỗi hay một xâu ký tự được đặt trong cặp dấu nháy kép (“ ”).

- Các chuỗi được lưu trữ trong bộ nhớ như là một dãy các ký tự liên tiếp và kết thúc bằng ký tự rỗng (NULL) có mã ASCII là 0.

Ví dụ: “Ngon ngu lap trinh C”

“Khoa CNTT-DHNT”

- Phân biệt: “A” và ‘A’

# Hằng xử lý trước biên dịch

- Các hằng có thể được xác lập trước khi biên dịch
  - Bản chất là tìm kiếm và thay thế
  - Thường được đặt tên với các chữ cái in hoa

Tìm từ “PI”, thay bằng 3.1415....

```
#include <stdio.h>

#define PI
#define DAYS_IN_WEEK
#define SUNDAY

int day = SUNDAY;
```

3.141592653590L  
7 ←  
0

Lưu ý: không  
có “=” và “;”

### 3. Biểu thức

- Biểu thức là sự kết hợp giữa các toán tử (operator) và các toán hạng (operand) theo đúng một trật tự nhất định.
- Mỗi toán hạng có thể là một hằng, một biến hoặc một biểu thức khác.
- Trong trường hợp, biểu thức có nhiều toán tử, ta dùng cặp dấu ngoặc đơn () để chỉ định toán tử nào được thực hiện trước.

Ví dụ: Biểu thức nghiệm của phương trình bậc hai:

$$(-b + \sqrt{\Delta}) / (2 * a)$$

Trong đó 2 là hằng; a, b, Delta là biến.

# Chương 2: Các toán tử

1. Toán tử gán
2. Toán tử số học
3. Toán tử quan hệ
4. Toán tử logic
5. Toán tử thao tác bit
6. Toán tử sizeof
7. Toán tử chọn theo điều kiện
8. Toán tử con trỏ
9. Toán tử dấu phẩy
10. Thứ tự ưu tiên của các phép toán
11. Hàm xuất dữ liệu
12. Hàm nhập dữ liệu

# 1. Toán tử gán:

- Cú pháp

<biến> = <biểu thức>

- Có thể sử dụng liên tiếp nhiều phép gán
- Giá trị được gán sẽ sẵn sàng cho lệnh kế tiếp

```
int i, j, k, l, m, n;  
  
i = j = k = l = m = n = 22;  
  
printf("%i\n", j = 93);
```

“n = 22” gán trước, lại gán “n” cho “m”, “m” cho “l”, ... → i, j, k, l, m, n đều nhận giá trị 22.

“j” được gán 93, giá trị 93 sẽ được in ra màn hình

# Toán tử gán (tiếp)

- Có thể sử dụng liên tiếp nhiều phép gán

```
int i, j, k, l, m, n;  
i = j = k = l = m = n = 11;  
  
printf("%i\n", j = 91);
```

“n = 11” gán trước, lại gán “n” cho “m”, “m” cho “l”, ... → i, j, k, l, m, n đều nhận giá trị 11.

“j” được gán 91, giá trị 91 sẽ được in ra màn hình

# Một số phép gán đặc biệt

- Các phép gán kết hợp toán tử khác:

`+=`

`-=`

`*=`

`/=`

`%=`

`&=`

`|=`

`^=`

`<<=`

`>>=`

- Tổng quát:

`<biến> <toán tử>= <biểu thức>`

tương đương:

`<biến> = <biến> <toán tử> <biểu thức>`

`a += 27;`

`a = a + 27;`

`f /= 9.2;`

`f = f / 9.2;`

`i *= j + 2;`

`i = i * (j + 2);`

## 2. Các phép toán số học

+	cộng
-	trừ
*	nhân
/	chia
%	chia lấy dư

### Lưu ý:

- “/” cho kết quả phụ thuộc vào kiểu của các toán hạng
- “%” không thực hiện được với các số thực

# Ví dụ về toán tử chia “/”

- Trình biên dịch dựa vào kiểu của các toán hạng để quyết định phép chia tương ứng

“i”, “j” kiểu int, “/” là phép chia lấy nguyên  
→ k nhận giá trị 1

“f”, “g” kiểu double, “/” là phép chia số thực  
→ h nhận giá trị 1.25

Phép chia nguyên, bắt  
kể “l” có kiểu double.  
Kết quả là 1.00000

```
int main(void)
{
    int     i = 5,    j = 4,    k;
    double f = 5.0,  g = 4.0, h, l;

    k = i / j;
    h = f / g;
    l = i / j;

    return 0;
}
```

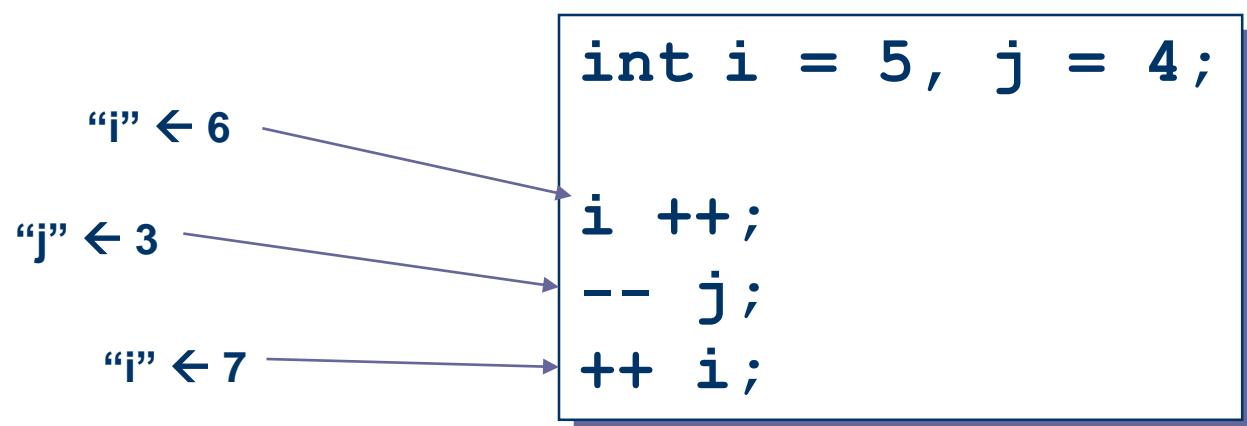
# Phép tăng (giảm) 1

- NNLT C có 2 toán tử đặc biệt hỗ trợ việc tăng (giảm) giá trị của một biến thay đổi 1 đơn vị

++      tăng 1

--      giảm 1

- Các toán tử này có thể đặt ở trước hoặc sau biến.



# Trước hay sau ?

Thứ tự thực hiện các toán tử ++ và -- phụ thuộc vào vị trí của chúng (trước hay sau) so với biến:

```
#include <stdio.h>

int main(void)
{
    int i, j = 5;
    i = ++j;
    printf("i=%d, j=%d\n", i, j);
    j = 5;
    i = j++;
    printf("i=%d, j=%d\n", i, j);

    return 0;
}
```

Tương đương:

1.  $j++;$
2.  $i = j;$

Tương đương:

1.  $i = j;$
2.  $j++;$

i=6, j=6  
i=5, j=6

# Chuyển kiểu/Ép kiểu (Type Casting)

- Chuyển kiểu làm thay đổi **tạm thời** kiểu của một biến trong một biểu thức.

Phép chia số nguyên  
được thực hiện, kết  
quả, 1, được đổi sang  
kiểu double, 1.00000

```
int main(void)
{
    int i = 5, j = 4;
    double f;

    f = (double)i / j;
    f = i / (double)j;
    f = (double)i / (double)j;
    f = (double)(i / j);

    return 0;
}
```

### 3. Các toán tử quan hệ (Relational Operators)

- NNLT C hỗ trợ các toán tử quan hệ:

< bé hơn

<= bé hơn hay bằng

> lớn hơn

>= lớn hơn hay bằng

== bằng

!= khác

- Tất cả đều cho kết quả **1** khi so sánh đúng và **0** trong trường hợp ngược lại.

## 4. Toán tử logic

- NNLT C hỗ trợ các toán tử logic:

**&&** và (and)

**||** hoặc (or)

**!** phủ định (not)

- Tất cả đều cho kết quả 1 hoặc 0 tương ứng

```
int i, j = 10, k = 28;  
  
i = ((j > 5) && (k < 100)) || (k > 24);
```

## 5. Toán tử thao tác bit (Bitwise Operators)

- Các toán tử trên bit chỉ có tác dụng trên các kiểu số nguyên:

&	And
	Or
^	Xor
~	Not
<<	Shift Left
>>	Shift Right

## 6. Toán tử sizeof

### sizeof (Obj)

- Cho biết kích thước của đối tượng theo số byte

```
#include <stdio.h>

int main(void)
{
    long    big;

    printf("\"big\" is %u bytes\n", sizeof(big));
    printf("a short is %u bytes\n", sizeof(short));
    printf("a double is %u bytes\n", sizeof (double));

    return 0;
}
```

"big" is 4 bytes  
a short is 2 bytes  
a double is 8 bytes

# 7. Toán tử chọn theo điều kiện

(điều kiện) ? BT1 : BT2

- Biểu thức nhận giá trị BT1 nếu điều kiện khác 0 (ĐÚNG), các trường hợp khác nhận giá trị BT2

```
int i, j = 100, k = -1;  
i = (j > k) ? j : k;
```

```
int i, j = 100, k = -1;  
i = (j < k) ? j : k;
```

Nếu  $j > k$   
     $i = j;$   
Ngược lại  
     $i = k;$

Nếu  $j < k$   
     $i = j;$   
Ngược lại  
     $i = k;$

## 8. Toán tử con trỏ

- + Một con trỏ là địa chỉ trong bộ nhớ của một biến. Một biến con trỏ là một biến được khai báo riêng để chứa một con trỏ đến một đối tượng của kiểu đã chỉ ra nó.
- + Có hai toán tử được sử dụng để thao tác với các con trỏ.
  - Toán tử thứ nhất là &, là một toán tử quy ước trả về địa chỉ bộ nhớ của hệ số của nó.

Ví dụ:  $p = \&n$

Đặt vào biến m địa chỉ bộ nhớ của biến count.

Chẳng hạn, biến n ở vị trí bộ nhớ 2000, giả sử n có giá trị là 100. Sau câu lệnh trên p sẽ nhận giá trị 2000.

- Toán tử thứ hai là \*, là một bổ sung cho &; đây là một toán tử quy ước trả về giá trị của biến được cấp phát tại địa chỉ theo sau đó.

Ví dụ:  $m = *p$

Sẽ đặt giá trị của n vào m. Bây giờ m sẽ có giá trị là 100 vì 100 được lưu trữ tại địa chỉ 2000.

## 9. Toán tử dấu phẩy ,

- Toán tử dấu , được sử dụng để kết hợp các biểu thức lại với nhau. Bên trái của toán tử dấu , luôn được xem là kiểu void. Điều đó có nghĩa là biểu thức bên phải trở thành giá trị của tổng các biểu thức được phân cách bởi dấu phẩy.
  - Ví dụ:  $x = (y=3,y+1);$
- Trước hết gán 3 cho y rồi gán 4 cho x. Cặp dấu ngoặc đơn là cần thiết vì toán tử dấu , có độ ưu tiên thấp hơn toán tử gán.

# 10. Độ ưu tiên của toán tử

- Thứ tự thực hiện các toán tử trong một biểu thức phụ thuộc vào *độ ưu tiên* của chúng.
- Có 16 mức ưu tiên.
- Thông thường, toán tử một ngôi có độ ưu tiên cao hơn toán tử hai ngôi.
- Các cặp dấu ngoặc đơn () thường được dùng để chỉ rõ thứ tự các toán tử.

```
#include <stdio.h>
int main(void)
{
    int j = 3 * 4 + 48 / 7;
    printf("j = %i\n", j);
    return 0;
}
```

j = 18

# Bảng thứ tự thực hiện các toán tử

Toán tử	Thứ tự (nếu cùng ĐURT)
() [] -> .	→
! ++ -- - + (cast) * & sizeof	←
* / %	→
+ -	→
<< >>	→
< <= >= >	→
== !=	→
&	→
	→
^	→
&&	→
	→
? :	←
= += -= *= /= %= ...	←
,	←

# Ví dụ

```
#include <stdio.h>

int main(void)
{
    int i = 0, j, k = 7, m = 5, n;
    j = m += 2;
    printf("j = %d\n", j);
    j = k++ > 7;
    printf("j = %d\n", j);
    j = i == 0 & k;
    printf("j = %d\n", j);
    n = !i > k >> 2;
    printf("n = %d\n", n);
    return 0;
}
```

# 11. Hàm xuất - printf

Xuất dữ liệu ra màn hình:

printf("Chuỗi định dạng ", Các biểu thức);

**printf (" %d-%d=%d\n" ,a,b, a - b) ;**

Các ký tự hằng được in nguyên văn

Các ký tự định dạng được thay bằng giá trị của biểu thức tương ứng:

%d: ký tự định dạng số nguyên kiểu int

Các ký tự điều khiển: \n – xuống dòng; \t – dấu tab;  
\\ – dấu \; \" – dấu “ ...

Thư viện: **stdio.h**

# 11. Hàm xuất – printf (tiếp)

%d	Xuất số nguyên
%[.số chữ số thập phân] f	Xuất số thực có <số chữ số thập phân> theo quy tắc làm tròn số.
%o	Xuất số nguyên hệ bát phân
%x	Xuất số nguyên hệ thập lục phân
%c	Xuất một ký tự
%s	Xuất chuỗi ký tự
%e hoặc %E hoặc %g hoặc %G	Xuất số nguyên dạng khoa học (nhân 10 mũ x)
<b>Ví dụ</b>	
%d	In ra số nguyên
%4d	In số nguyên tối đa 4 ký số, nếu số cần in nhiều hơn 4 ký số thì in hết
%f	In số thực
%6f	In số thực tối đa 6 ký số (tính luôn dấu chấm), nếu số cần in nhiều hơn 6 ký số thì in hết
%.3f	In số thực có 3 số lẻ, nếu số cần in có nhiều hơn 3 số lẻ thì làm tròn.

# 12. Hàm nhập - scanf

Nhập dữ liệu từ bàn phím

```
scanf("Chuỗi định dạng", địa chỉ của các biến);
```

**scanf ("%d %d", &a, &b) ;**

- Trong chuỗi định dạng chỉ có ký tự định dạng và khoảng trắng.
- Dữ liệu phải được nhập vào các biến.
- Trước tên biến phải ghi dấu & - toán tử địa chỉ. Nếu không có toán tử địa chỉ, giá trị của biến sẽ không được cập nhật
- Thư viện: **stdio.h**

# 12. Hàm nhập – scanf(tiếp)

Định dạng	Ý nghĩa
%[số ký số]d	Nhập số nguyên có tối đa <số ký số>
%[số ký số] f	Nhập số thực có tối đa <số ký số> tính cả dấu chấm
%c	Nhập một ký tự
<b>Ví dụ:</b>	
%d	Nhập số nguyên
%4d	Nhập số nguyên tối đa 4 ký số, nếu nhập nhiều hơn 4 ký số thì chỉ nhận được 4 ký số đầu tiên
%f	Nhập số thực
%6f	Nhập số thực tối đa 6 ký số (tính luôn dấu chấm), nếu nhập nhiều hơn 6 ký số thì chỉ nhận được 6 ký số đầu tiên (hoặc 5 ký số với dấu chấm)

# Chương 3: Các cấu trúc điều khiển

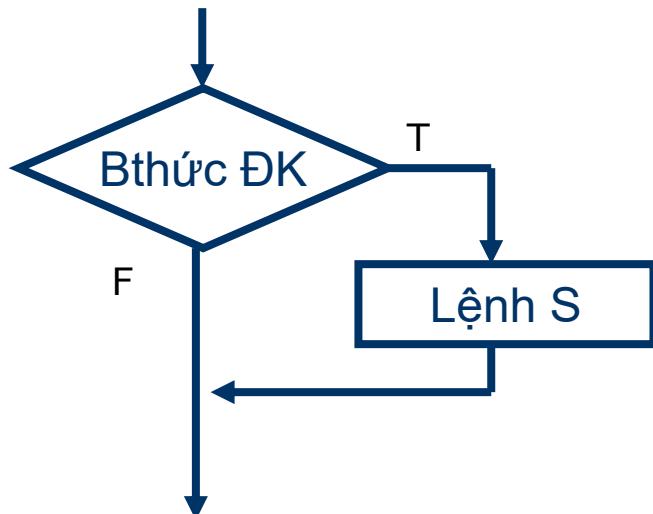
1. Lệnh if
2. Lệnh switch
3. Lệnh for
4. Lệnh while
5. Lệnh do .. While

# 1. Lệnh if

- Dùng để thực hiện hay không một phát biểu theo một điều kiện.

- Dạng 1:

if (<Bthức ĐK>) <Lệnh S>;



Chỉ có 1  
phát biểu  
trong thân  
của if

- Ví dụ

if (delta > 0)

{

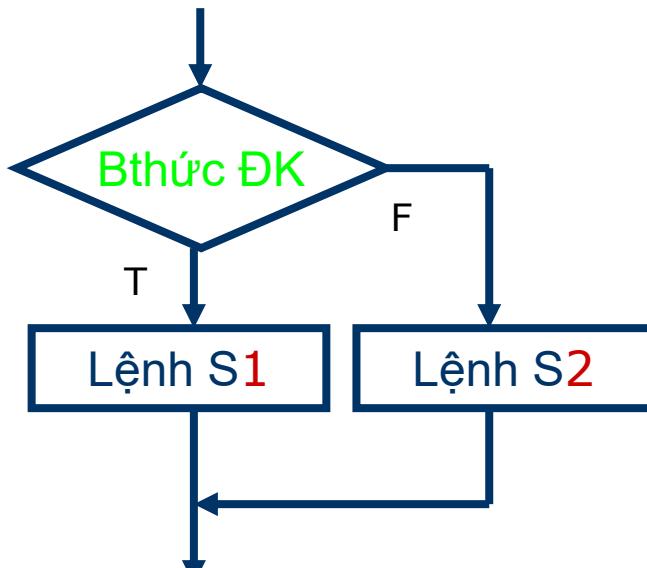
$x1 = (-b + \sqrt{\text{delta}})/2/a;$

$x2 = (-b - \sqrt{\text{delta}})/2/a;$

}

# Dạng 2 của lệnh if

- Dùng để chọn lựa phát biểu nào sẽ được thực hiện giữa 2 phát biểu.
- Cú pháp:**if (< Bthức ĐK >) <Lệnh S1> ; else <Lệnh S2> ;**



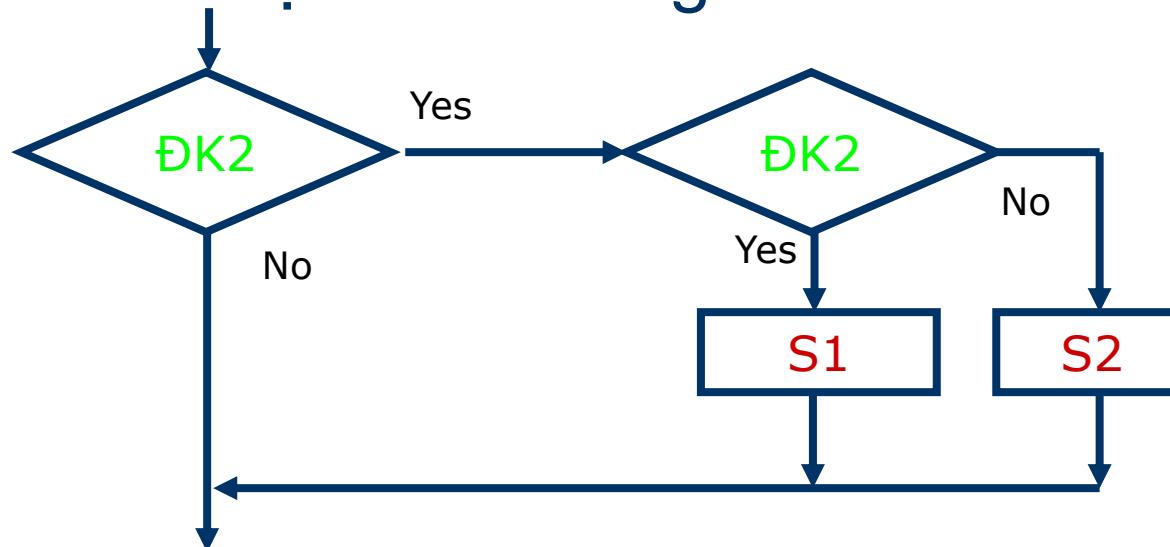
```
#include <stdio.h>
#include <conio.h>
int main ()
{
    float a;
    printf("Nhập a = "); scanf("%f",&a);
    if (a !=0 )
        printf("Nghịch đảo của %f là %f",a,1/a);
    else
        printf("Không thể tìm nghịch đảo của a");
    getch();
    return 0;
}
```

# Trường hợp đặc biệt

- Xét phát biểu sau:
- **if (<ĐK1>) if (<ĐK2>) <S1>; else <S2>;**

else ? else

- else sẽ thuộc về if nào gần 1 chưa có else



## 2. Lệnh switch

- Dùng để chọn một trong số những phát biểu để thực hiện tùy theo giá trị của biểu thức chọn.
- Các giá trị case: chỉ ra các trường hợp phân nhánh.
- Các giá trị case là một hay nhiều giá trị rời rạc sau là dấu : và một phát biểu tương ứng, kết thúc là break.

Cú pháp:

```
switch (<Biểu thức>)
```

```
{
```

```
    case <giá trị 1>: <LệnhS1>; break;
```

```
    ...
```

```
    case <giá trị n>: <Lệnh Sn>; break;
```

```
[default : <lệnh mặc định Sn+1>]
```

```
}
```



# Ví dụ 1

Nhập vào một số nguyên, chia số nguyên này cho 2 lấy phần dư. Kiểm tra nếu phần dư bằng 0 thì in ra thông báo “số chẵn”, nếu số dư bằng 1 thì in thông báo “số lẻ”.

```
#include <stdio.h>
#include<conio.h>
int main ()
{ int songuyen, phandu;
clrscr();
printf("\n Nhập vào số nguyên ");
scanf("%d",&songuyen); phandu=(songuyen % 2);
switch(phandu)
{
    case 0: printf("%d là số chẵn ",songuyen); break;
    case 1: printf("%d là số lẻ ",songuyen); break;
}
getch();
return 0;
}
```

# Ví dụ 2

Ví dụ 2: Nhập vào 2 số nguyên và 1 phép toán.

- Nếu phép toán là '+', '-' , '\*' thì in ra kết quả là tổng, hiệu, tích của 2 số.
- Nếu phép toán là '/' thì kiểm tra xem số thứ 2 có khác không hay không? Nếu khác không thì in ra thương của chúng, ngược lại thì in ra thông báo "khong chia cho 0".

```
#include <stdio.h>
#include<conio.h>
int main ()
{ int so1, so2; float thuong; char pheptoan;
    printf("\n Nhập vào 2 số nguyên "); scanf("%d%d",&so1,&so2);
    fflush(stdin); /*Xóa ký tự enter trong vùng đệm trước khi nhập phép toán */
    printf("\n Nhập vào phép toán "); scanf("%c",&pheptoan);
    switch(pheptoan)
    {
        case '+': printf("\n %d + %d =%d",so1, so2, so1+so2); break;
        case '-': printf("\n %d - %d =%d",so1, so2, so1-so2); break;
        case '*': printf("\n %d * %d =%d",so1, so2, so1*so2); break;
        case '/': if (so2!=0)
                    { thuong=float(so1)/float(so2);
                      printf("\n %d / %d =%f", so1, so2, thuong); }
                    else printf("Không chia được cho 0"); break;
        default : printf("\n Chưa hỗ trợ phép toán %c", pheptoan); break;
    }
    getch();
    return 0;
}
```

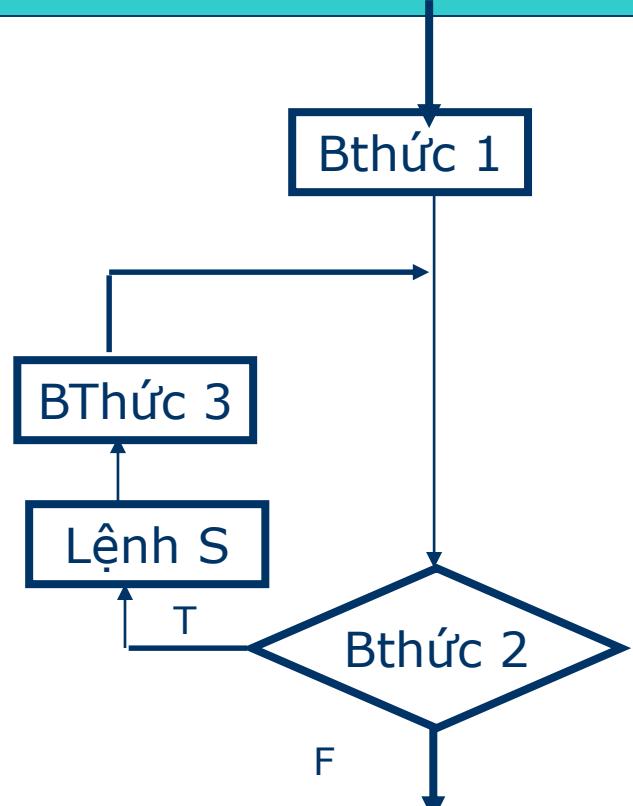
### 3. Lệnh for

Lệnh for cho phép lặp lại các lệnh cho đến khi Biểu thức điều kiện 2 là sai

Cú pháp: `for (<biểuthức 1>;<biểuthức 2>;<biểuthức 3>) <lệnh S>`

Ví dụ: Viết chương trình nhập vào một số nguyên n. Tính tổng của các số nguyên từ 1 đến n.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    unsigned int n,i,tong;
    printf("\n Nhập vào số n:");scanf("%d",&n);
    tong=0;
    for (i=1; i<=n; i++) tong+=i;
    printf("\n Tổng từ 1 đến %d =%d ",n,tong);
    getch();
    return 0;
}
```



# 4. Lệnh while

Dùng để lặp lại một công việc nào đó cho đến khi điều kiện sai.

Cú pháp

*while (<Biểu thức ĐK>) <Lệnh S>*

while kiểm tra điều kiện trước rồi mới thực hiện lệnh S.

Số lặp lặp là không biết trước.

Số lần lặp tối thiểu là 0 và tối đa là không xác định.

Chú ý: Trong thân của while phải có ít nhất một phát biểu có khả năng thay đổi giá trị của điều kiện. Nếu không sẽ lặp vô tận (infinite loop)

- Ví dụ:

gt=1; i=1;

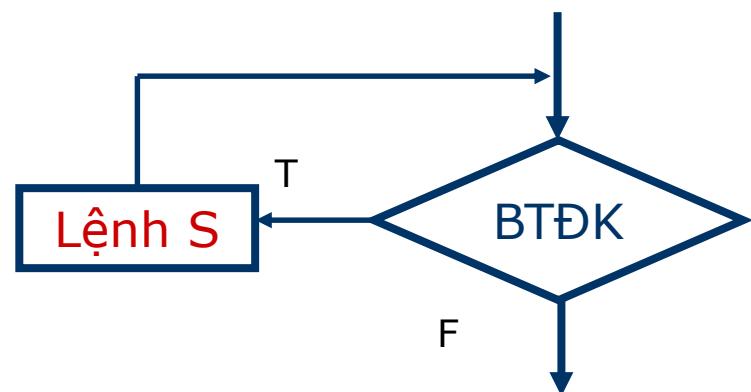
while (i<n)

{

i++;

gt=gt\*i;

}



# Ví dụ về lệnh while

Ví dụ 2: Viết chương trình nhập vào một số nguyên n. Tính tổng của các số nguyên từ 1 đến n.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    unsigned int n,i,tong;
    printf("\n Nhập vào số nguyên dương n:"); scanf("%d",&n);
    tong=0;
    i=1;
    while (i<=n)
    {
        tong+=i;
        i++;
    }
    printf("\n Tổng từ 1 đến %d =%d ",n,tong);
    getch();
    return 0;
}
```

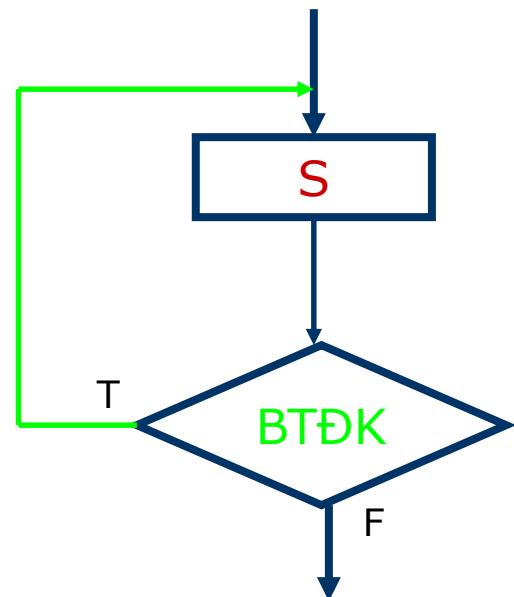
# 5. Lệnh do .. while

Vòng lặp do ... while dùng để lặp lại một công việc nào đó khi điều kiện còn đúng.

Cú pháp:

**do <Lệnh S> while (<Biểu thức điều kiện>)**

- Thực hiện xong lệnh S mới kiểm tra điều kiện.
- Số lần lặp là không biết trước.
- Số lần lặp tối thiểu là 1 và tối đa là không xác định.
- Chú ý: Trong thân của do .. While phải có ít nhất một phát biểu có khả năng thay đổi giá trị của điều kiện. Nếu không sẽ lặp vô tận (infinite loop)



# Ví dụ về lệnh do .. while

Viết chương trình nhập vào một số nguyên n. Tính tổng của các số nguyên từ 1 đến n.

```
#include <stdio.h>
#include<conio.h>
int main ()
{
    unsigned int n,i,tong;
    printf("\n Nhập vào số nguyên dương n:"); scanf("%d",&n);
    tong=0;
    i=1;
    do
    {
        tong+=i;
        i++;
    } while (i<=n);
    printf("\n Tổng từ 1 đến %d =%d ",n,tong);
    getch();
    return 0;
}
```

# 6. CÁC CÂU LỆNH ĐẶC BIỆT

## 1. Lệnh break

Cú pháp: break

Dùng để thoát khỏi vòng lặp. Khi gặp câu lệnh này trong vòng lặp, chương trình sẽ thoát ra khỏi vòng lặp và chỉ đến câu lệnh liền sau nó. Nếu nhiều vòng lặp thì break sẽ thoát ra khỏi vòng lặp gần nhất. Ngoài ra, break còn được dùng trong cấu trúc lựa chọn switch.

## 2. Lệnh continue

Cú pháp: continue

- Khi gặp lệnh này trong các vòng lặp, chương trình sẽ bỏ qua phần còn lại trong vòng lặp và tiếp tục thực hiện lần lặp tiếp theo.
- Đối với lệnh for, biểu thức 3 sẽ được tính trị và quay lại bước 2.
- Đối với lệnh while, do while; biểu thức điều kiện sẽ được tính và xét xem có thể tiếp tục thực hiện <Lệnh S> nữa hay không? (dựa vào kết quả của biểu thức điều kiện).

Ví dụ:

```
while (x != y)
{
    ...
    if (x==a) continue;
    b+=6;

    ...
    if (y==b) break;
    ...
}
```

# Chương 4: Các hàm trong C

## 1. Khái niệm hàm trong C

- Tại sao phải dùng chương trình con:
  - Có công việc cần phải được thực hiện tại nhiều nơi trong chương trình => tách công việc đó thành chương trình con
  - Phân đoạn, module chương trình để thuận tiện trong quản lý, trình bày và phát triển.
- Trong C, chương trình con được gọi là hàm. Hàm trong C có thể trả về kết quả thông qua tên hàm hay có thể không trả về kết quả.
- Hàm có hai loại: hàm chuẩn (hàm được trình biên dịch C viết sẵn) và hàm tự định nghĩa.
- Một hàm khi được định nghĩa thì có thể được gọi trong chương trình.
- Trong C, hàm main() được gọi thực hiện đầu tiên

# Ví dụ

Viết hàm main dùng để nhập vào 2 số nguyên a,b và in ra màn hình số lớn trong 2 số đã nhập

```
#include <stdio.h>
#include <conio.h>
int max(int a, int b);

int main()
{
    int a, b, c;
    printf("\n Nhập vào 3 số a, b,c ");
    scanf("%d%d%d",&a,&b,&c);
    printf("\n Số lớn là %d",max(a, max(b,c)));
    getch();
    return 0;
}

int max(int a, int b)
{
    return (a>b) ? a:b;
}
```

# Ghi chú

- Dòng int max(int a, int b); gọi là Prototype của hàm, qui định kiểu trả về của hàm, số lượng tham số và kiểu của chúng.
- Các Prototype của các hàm sẵn có chứa trong các tập tin \*.h
- Hàm có thể được gọi bởi hàm main(), hoặc từ một hàm khác hoặc chính nó(đệ qui).
- Hàm có thể có tham số hoặc không.
- Hàm chỉ có 1 điểm vào (lệnh đầu tiên của hàm) nhưng có thể có nhiều điểm ra (lệnh return).
- Một hàm có thể được viết ngay trong văn bản chương trình (như trên), hoặc được viết trong tập tin khác và đưa vào chương trình bằng chỉ thị #include, hoặc được biên dịch riêng rẽ và kết nối lại.
- Khác với ngôn ngữ lập trình Pascal:
  - Ngôn ngữ C không có khái niệm thủ tục (thật ra thủ tục không khác hàm, ở thủ tục không quan tâm đến giá trị trả về)
  - Không cho phép các hàm lồng vào nhau

## 2. Hàm thư viện/hàm chuẩn

- Hàm thư viện là những hàm đã được định nghĩa sẵn trong một thư viện nào đó, muốn sử dụng các hàm thư viện thì phải khai báo thư viện trước khi sử dụng bằng lệnh #include <tên thư viện.h>

### **Ý nghĩa của một số thư viện thường dùng:**

1. stdio.h : Thư viện chứa các hàm vào/ ra chuẩn (standard input/output). Gồm các hàm printf(), scanf(), getc(), putc(), gets(), puts(), fflush(), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw()...
2. conio.h : Thư viện chứa các hàm vào ra trong chế độ DOS (DOS console). Gồm các hàm clrscr(), getch(), getche(), getpass(), cgets(), cputs(), putch(), clreol(),...
3. math.h: Thư viện chứa các hàm tính toán gồm các hàm abs(), sqrt(), log(). log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow(), exp(),...
4. alloc.h: Thư viện chứa các hàm liên quan đến việc quản lý bộ nhớ. Gồm các hàm calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(), ...
5. io.h: Thư viện chứa các hàm vào ra cấp thấp. Gồm các hàm open(), \_open(), read(), \_read(), close(), \_close(), creat(), \_creat(), creatnew(), eof(), filelength(), lock(),...
6. graphics.h: Thư viện chứa các hàm liên quan đến đồ họa. Gồm initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(), ...

...

Muốn sử dụng các hàm thư viện thì ta phải xem cú pháp của các hàm và sử dụng theo đúng cú pháp (xem trong phần trợ giúp của Turbo C).

### 3. Hàm của người sử dụng

- Hàm người dùng là những hàm do người lập trình tự tạo ra nhằm đáp ứng nhu cầu xử lý của mình.

Cấu trúc của hàm:

```
<kiểu kết quả> <Tên hàm> ([<kiểu> <tham số>][,<kiểu><tham số>][...])  
{  
    [Khai báo]  
    <câu lệnh thực hiện hàm>  
    [return <Biểu thức>;]  
}
```

# Truyền Bằng Trị - Tham biến - Tham Chiếu

Ví dụ: Viết chương trình hoán vị 2 phần tử

```
#include<stdio.h>

// Truyền bằng tham trị
1 void Swap1 (int x, int y)
2 {
3     int temp = x;
4     x = y;
5     y = temp;
6 }

// Truyền bằng tham biến (con trỏ)
7 void Swap2 (int *x, int *y)
8 {
9     int temp = *x;
10    *x = *y;
11    *y = temp;
12 }

// Truyền bằng tham chiếu
13 void Swap3 (int &x, int &y)
14 {
15     int temp = x;
16     x = y;
17     y = temp;
18 }
```

```
int main()
{
    int m=12; n=28;
    Swap1(m,n);
    printf("m=%d n=%d\n",m,n);
    Swap2(&m,&n);
    printf("m=%d n=%d\n",m,n);
    Swap3(m,n);
    printf("m=%d n=%d\n",m,n);
    return 0;
}
```



# 4. Tham trị và tham biến

Mặc nhiên, việc truyền tham số cho hàm trong C là truyền theo giá trị; nghĩa là các giá trị thực (tham số thực) không bị thay đổi giá trị khi truyền cho các tham số hình thức

Ví dụ 1: Giả sử muốn in ra các, mỗi dòng gồm 50 ký tự nào đó. Để đơn giản ta viết hàm, hàm này sẽ in ra trên một dòng 50 ký tự cho trước.

```
#include <stdio.h>
#include <conio.h>
void InKT(char ch)
{ int i;
    for(i=1;i<=50;i++) printf("%c",ch);
} printf("\n");

int main()
{
    char c = 'A';
    InKT('*'); /* In ra 50 dấu * */
    InKT('+');
    InKT(c);
    return 0;
}
```

Lưu ý:

- Trong hàm InKT ở trên, biến ch gọi là tham số hình thức được truyền bằng giá trị (gọi là tham trị của hàm). Các tham trị của hàm coi như là một biến cục bộ trong hàm và chúng được sử dụng như là dữ liệu đầu vào của hàm.
- Khi chương trình con được gọi để thi hành, tham trị được cấp ô nhớ và nhận giá trị là bản sao giá trị của tham số thực.
- Việc thay đổi giá trị của chúng không có ý nghĩa gì đối với bên ngoài hàm, không ảnh hưởng đến chương trình chính, nghĩa là không làm ảnh hưởng đến tham số thực tương ứng.

# 5. HÀM ĐỆ QUY

Định nghĩa: Một hàm được gọi là đệ quy nếu bên trong thân hàm có lệnh gọi đến chính nó.

Ví dụ: Định nghĩa giai thừa của một số nguyên dương n như sau:

$$n! = 1 * 2 * 3 * \dots * (n-1) * n = (n-1)! * n \text{ (với } 0!=1)$$

Như vậy, để tính  $n!$  ta thấy nếu  $n=0$  thì  $n!=1$  ngược lại thì  $n!=n * (n-1)!$

Với định nghĩa trên thì hàm đệ quy tính  $n!$  được viết:

```
#include <stdio.h>
#include <conio.h>
/*Hàm tính n! bằng đệ quy*/
unsigned int giaithua_dequy(int n)
{
    if (n==0) return 1;
    else return n*giaithua_dequy(n-1);
}

/*Hàm tính n! không đệ quy*/
unsigned int giaithua_khongdequy(int n)
{
    unsigned int kq,i;
    kq=1;
    for (i=2;i<=n;i++) kq=kq*i;
    return kq;
}
```

```
int main()
{
    int n;
    printf("\n Nhập số n cần tính giai thừa "); scanf("%d",&n);
    printf("\nGọi ham de quy: %d !=%u",n,giaithua_dequy(n));
    printf("\nGọi ham khong de quy: %d != %u",
           n,giaithua_khongdequy(n));
    getch();
    return 0;
}
```

## Đặc điểm cần lưu ý khi viết hàm đệ quy

- Hàm đệ quy phải có 2 phần:
  - + Phần dừng hay phải có trường hợp nguyên tố. Trong ví dụ ở trên thì trường hợp  $n=0$  là trường hợp nguyên tố.
  - + Phần đệ quy: là phần có gọi lại hàm đang được định nghĩa. Trong ví dụ trên thì phần đệ quy là  $n>0$  thì  $n! = n * (n-1)!$
- Sử dụng hàm đệ quy trong chương trình sẽ làm chương trình dễ đọc, dễ hiểu và vấn đề được nêu bật rõ ràng hơn. Tuy nhiên trong đa số trường hợp thì hàm đệ quy tốn bộ nhớ nhiều hơn và tốc độ thực hiện chương trình chậm hơn không đệ quy.
- Tùy từng bài có cụ thể mà người lập trình quyết định có nên dùng đệ quy hay không (có những trường hợp không dùng đệ quy thì không giải quyết được bài toán).

# 6. Tham số của hàm main()

- Là 2 tham số : argc và argv
- Tham số argc là số nguyên chỉ tham số trên dòng lệnh, có giá trị nhỏ nhất =1, vì bản thân tên chương trình là tham số thứ nhất
- Tham số argv là mảng các con trỏ, trỏ đến các tham số trên dòng lệnh: char \*argv[ ];
  - argv[0]: chứa địa chỉ của tên chương trình
  - argv[1]: chứa địa chỉ của tham số thứ nhất
  - argv[2]: chứa địa chỉ của tham số thứ hai

Ví dụ: Chương trình sau đã được biên dịch thành MYPRO.EXE, nếu nhập trên dòng lệnh  
MYPRO thì có dòng nhắc nhở, nếu nhập MYPRO LAN thì Chao ban LAN

```
#include <stdio.h>
main(int argc, char *argv[])
{
    if (argc !=2) printf("Phai nhap Ten");
    else printf("Chao ban %s\n",argv[1]);
}
```

# Chương 5: Dữ liệu có cấu trúc

## 1. Kiểu mảng:

- Mảng là một tập hợp các phần tử cố định có cùng một kiểu, gọi là kiểu phần tử. Kiểu phần tử có thể là có các kiểu bất kỳ: ký tự, số, chuỗi ký tự...; cũng có khi ta sử dụng kiểu mảng để làm kiểu phần tử cho một mảng (trong trường hợp này ta gọi là mảng của mảng hay mảng nhiều chiều).
- Có thể chia mảng làm 2 loại: mảng 1 chiều và mảng nhiều chiều.
- **Khai báo mảng với số phần tử xác định**  
Cú pháp: <Kiểu> <Tên mảng> <[số phần tử]>

Ý nghĩa:

- **Tên mảng**: đây là một cái tên đặt đúng theo quy tắc đặt tên của danh biều. Tên này cũng mang ý nghĩa là tên biến mảng.
- **Số phần tử**: là một hằng số nguyên, cho biết số lượng phần tử tối đa trong mảng là bao nhiêu (nói khác là kích thước của mảng).
- **Kiểu**: mỗi phần tử của mảng có dữ liệu thuộc kiểu gì.
- Khi khai báo một biến mảng gồm có **số phần tử** phần tử, phần tử thứ nhất là **tên mảng [0]**, phần tử cuối cùng là **tên mảng[số phần tử - 1]**

# 1. Kiểu mảng(tiếp)

Ví dụ: int a[10]; /\* Khai báo biến mảng a gồm 10 phần tử , phần tử thứ nhất là a[0], phần tử cuối cùng là a[9].\*/

Có thể coi mảng a là một dãy liên tiếp các phần tử trong bộ nhớ như sau:

Vị trí	0	1	2	3	4	5	6	7	8	9
Tên phần tử	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

## 2. Khai báo mảng với số phần tử không xác định

Cú pháp: <Kiểu> <Tên mảng> <[]>

Khi khai báo, không cho biết rõ số phần tử của mảng, kiểu khai báo này thường được áp dụng trong các trường hợp: vừa khai báo vừa gán giá trị, khai báo mảng là tham số hình thức của hàm.

### a. Vừa khai báo vừa gán giá trị

Cú pháp: <Kiểu> <Tên mảng> []= {Các giá trị phân cách bởi dấu ,}

Nếu vừa khai báo vừa gán giá trị thì mặc nhiên C sẽ hiểu số phần tử của mảng là số giá trị mà chúng ta gán cho mảng trong cặp dấu {}.

### b. Khai báo mảng là tham số hình thức của hàm, trong trường hợp này ta không cần chỉ định số phần tử của mảng là bao nhiêu.

# 1. Kiểu mảng(tiếp)

- Truy xuất từng phần tử của mảng
  - + Mỗi phần tử của mảng được truy xuất thông qua Tên biến mảng theo sau là chỉ số nằm trong cặp dấu ngoặc vuông []. Chẳng hạn a[0] là phần tử đầu tiên của mảng a được khai báo ở trên. Chỉ số của phần tử mảng là một biểu thức mà giá trị là kiểu số nguyên.
  - + Với cách truy xuất theo kiểu này, Tên biến mảng[Chỉ số] có thể coi như là một biến có kiểu dữ liệu là kiểu được chỉ ra trong khai báo biến mảng.

Ví dụ 2: Vừa khai báo vừa gán trị cho 1 mảng 1 chiều các số nguyên. In mảng số nguyên này lên màn hình.

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int n,i,j,tam;
    int dayso[]={66,65,69,68,67,70};
    n=sizeof(dayso)/sizeof(int); /*Lấy số phần tử*/
    printf("\n Nguồn dữ liệu của mảng ");
    for (i=0;i<n;i++) printf("%d ",dayso[i]);
    return 0;
}
```

# 1. Kiểu mảng(tiếp)

Ví dụ 4: Nhập vào một dãy n số và sắp xếp các số theo thứ tự tăng.

```
#include<conio.h>
#include<stdio.h>
void Nhap(int a[],int N)
{ int i;
  for(i=0; i< N; i++)
  {
    printf("Phan tu thu %d: ",i);
    scanf("%d",&a[i]);
  }
}

void InMang(int a[], int N)
{ int i;
  for (i=0; i<N;i++) printf("%d ",a[i]);
  printf("\n");
}

void SapXep(int a[], int N)
{ int t,i;
  for(i=0;i<N-1;i++)
    for(int j=i+1;j<N;j++)
      if (a[i]>a[j])
        { t=a[i]; a[i]=a[j]; a[j]=t; }
```

```
int main()
{ int b[20], N;
  printf("So phan tu cua mang N=");
  scanf("%d",&N);
  Nhap(b,N);
  printf("Mang vua nhap: ");
  InMang(b,N);
  SapXep(b,N); /* Gọi hàm sắp
  xếp*/
  printf("Mang sau khi sap xep: ");
  InMang(b,N);
  getch();
  return 0;
}
```

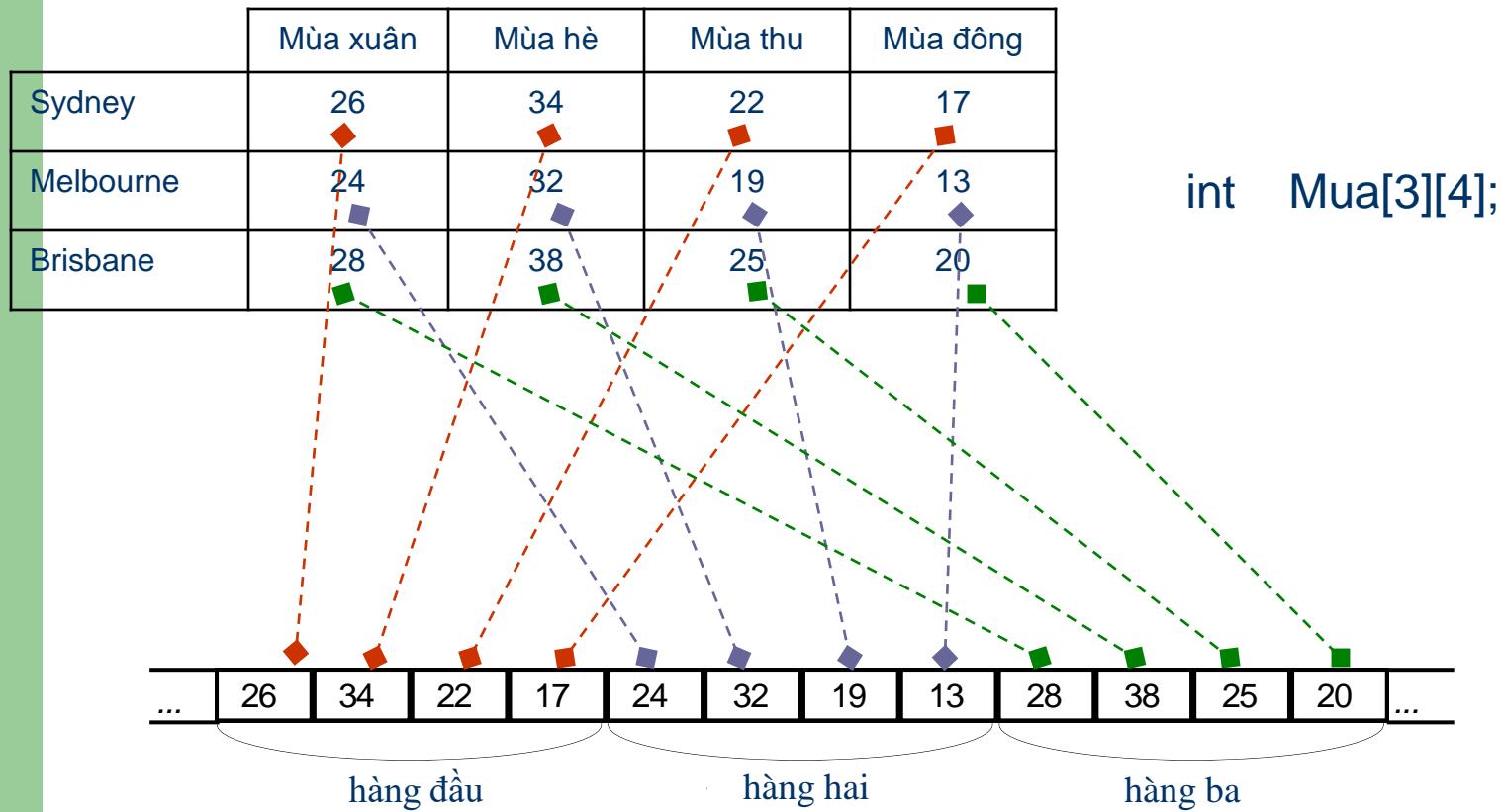
# 1. Kiểu mảng(tiếp)

Ví dụ: Viết chương trình cho phép nhập 2 ma trận a, b có m dòng n cột, thực hiện phép toán cộng hai ma trận a,b và in ma trận kết quả lên màn hình.

```
#include<conio.h>
#include<stdio.h>
void Nhap(int a[][10],int M,int N)
{ int i,j;
    for(i=0;i<M;i++)
        for(j=0; j<N; j++){
            printf("Phan tu o dong %d cot %d: ",i,j);
            scanf("%d",&a[i][j]);
        }
}
void InMaTran(int a[][10], int M, int N)
{ int i,j;
    for(i=0;i<M;i++){
        for(j=0; j< N; j++) printf("%d ",a[i][j]);
        printf("\n");
    }
}
/* Cong 2 ma tran A & B ket qua la ma tran C*/
void CongMaTran(int a[][10],int b[][10],int M,int N,int c[][10])
{ int i,j;
    for(i=0;i<M;i++)
        for(j=0; j<N; j++) c[i][j]=a[i][j]+b[i][j];
}
```

```
int main()
{ int a[10][10], b[10][10], M, N;
    int c[10][10];/* Ma tran tong*/
    printf("So dong M= "); scanf("%d",&M);
    printf("So cot M= "); scanf("%d",&N);
    printf("Nhap ma tran A\n"); Nhap(a,M,N);
    printf("Nhap ma tran B\n"); Nhap(b,M,N);
    printf("Ma tran A: \n"); InMaTran(a,M,N);
    printf("Ma tran B: \n"); InMaTran(b,M,N);
    CongMaTran(a,b,M,N,c);
    printf("Ma tran tong C:\n");
    InMaTran(c,M,N);
    getch();
    return 0;
}
```

# Mảng nhiều chiều



Cách tổ chức trong bộ nhớ

## 2. Kiểu chuỗi ký tự

- Kiểu chuỗi ký tự là một trường hợp đặc biệt của mảng các ký tự. Chuỗi ký tự kết thúc bằng ký tự có mã ASCII là 0 ('\0')
- Do đó, char[10]; //lưu trữ tối đa 9 ký tự

Ghi chú:

- + Khi áp dụng các phép toán trên chuỗi, ta phải sử dụng các hàm (string.h)

Ví dụ:-Hàm strcpy(chuỗi1,chuỗi 2):  
chép chuỗi 2 vào chuỗi 1

- Hàm strcmp(chuỗi1,chuỗi2):  
so sánh 2 chuỗi trả về 0 nếu 2 chuỗi bằng nhau ....

Ví dụ: Nhập vào một chuỗi và hiển thị trên màn hình chuỗi vừa nhập.

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
int main()
{   char Ten[12];
    printf("Nhập chuỗi: ");gets(Ten);
    printf("Chuỗi vừa nhập: ");puts(Ten);
    getch();
    return 0;
}
```

### 3. Kiểu con trỏ

Các biến trước đây đều là biến có kích thước và kiểu dữ liệu xác định. Gọi các biến kiểu này là biến tĩnh. Khi khai báo biến tĩnh, các ô nhớ sẽ được cấp phát mà không biết trong quá trình chạy chương trình có sử dụng hết chúng hay không.

Các biến tĩnh dạng này sẽ tồn tại trong suốt thời gian thực thi chương trình dù có những biến mà chương trình chỉ sử dụng 1 lần rồi bỏ. Các hạn chế về biến tĩnh:

- Cấp phát ô nhớ dư, gây ra lãng phí ô nhớ.
- Cấp phát ô nhớ thiếu, chương trình thực thi bị lỗi.

Để giải quyết những hạn chế trên, ngôn ngữ C cung cấp cho ta một loại biến đặc biệt gọi là biến động với các đặc điểm sau:

- Chỉ phát sinh trong quá trình chạy chương trình chứ không phát sinh lúc bắt đầu chương trình.
- Khi chạy chương trình, kích thước của biến, vùng nhớ và địa chỉ vùng nhớ được cấp phát cho biến có thể thay đổi.
- Sau khi sử dụng xong có thể giải phóng để tiết kiệm chỗ trong bộ nhớ.

Vì thế, ngôn ngữ C lại cung cấp cho ta một loại biến đặc biệt nữa để khắc phục tình trạng này, đó là biến con trỏ (pointer) với các đặc điểm:

- Biến con trỏ không chứa dữ liệu mà chỉ chứa địa chỉ của dữ liệu hay chứa địa chỉ của ô nhớ.
- Kích thước của biến con trỏ không phụ thuộc vào kiểu dữ liệu, luôn có kích thước cố định là 2 bytes nếu trong cùng 1 đoạn và 4 bytes nếu khác đoạn.

### 3. Kiểu con trỏ(tiếp)

#### a. Khai báo biến con trỏ

Cú pháp: <Kiểu> \* <biến con trỏ>

Ví dụ 1: Khai báo 2 biến a,b có kiểu int và 2 biến pa, pb là 2 biến con trỏ kiểu int.  
int a, b, \*pa, \*pb;

Ví dụ 2: Khai báo biến f kiểu float và biến pf là con trỏ float  
float f, \*pf;

#### b. Các phép toán trên con trỏ

+ Phép gán con trỏ: Hai con trỏ cùng kiểu có thể gán cho nhau.

Ví dụ: int a, \*p, \*a ; float \*f;

a = 5 ;

p = &a ;

q = p ; /\* đúng \*/

f = p ; /\* sai do khác kiểu \*/

+ Cộng, trừ con trỏ với một số nguyên: Có thể cộng (+), trừ (-) 1 con trỏ với 1 số nguyên N nào đó; kết quả trả về là 1 con trỏ. Con trỏ này chỉ đến vùng nhớ cách vùng nhớ của con trỏ hiện tại N phần tử.

- Đơn vị tăng hay giảm của con trỏ có kích thước của biến được trỏ đến

# 4. Con trỏ được dùng như mảng.

Ví dụ: Cho 1 mảng 1 chiều các số nguyên a có 5 phần tử, truy cập các phần tử theo kiểu mảng và theo kiểu con trỏ.

```
#include <stdio.h>
#include <conio.h>
/* Nhập mảng bình thường*/
void NhapMang(int a[], int N)
{
    int i;
    for(i=0;i<N;i++)
    {
        printf("Phan tu thu %d: ",i);
        scanf("%d",&a[i]);
    }
}
/* Nhập mảng theo dạng con trỏ*/
void NhapContro(int a[], int N)
{
    int i;
    for(i=0;i<N;i++)
    {
        printf("Phan tu thu %d: ",i);
        scanf("%d",a+i);
    }
}
```

```
int main()
{
    int a[20],N,i;
    printf("So phan tu N= ");scanf("%d",&N);
    NhapMang(a,N);
    printf("Truy cap theo kieu mang: ");
    for(i=0;i<N;i++) printf("%d ",a[i]);
    printf("\nTruy cap theo kieu con tro: ");
    for(i=0;i<N;i++) printf("%d ",*(a+i));
    getch();
    return 0;
}
```

# 5. Con trỏ và cấp phát động

## a. Hàm cấp phát:

**void \*malloc(size\_t size):** Cấp phát vùng nhớ có kích thước là size.

**void \*calloc(size\_t nitems, size\_t size):** Cấp phát vùng nhớ có kích thước là nitems\*size.

Ví dụ: Giả sử ta có khai báo:

```
int a, *pa, *pb; pa = (int*)malloc(sizeof(int)); /* Cấp phát vùng nhớ có kích  
thước bằng với  
của một số nguyên */
```

```
pb= (int*)calloc(10, sizeof(int)); /* Cấp phát vùng nhớ có thể chứa được 10 số  
nguyên*/
```

**Lưu ý:** Khi sử dụng hàm malloc() hay calloc(), ta phải ép kiểu vì nguyên mẫu các  
hàm này trả về con trỏ kiểu void.

**b. Thu hồi vùng nhớ:** Một vùng nhớ đã cấp phát cho biến động do biến con trỏ giữ  
địa chỉ, khi không còn sử dụng nữa, ta sẽ thu hồi lại vùng nhớ này nhờ hàm  
free().

**Cú pháp: void free(void \*block)**

**Ý nghĩa:** Giải phóng vùng nhớ được quản lý bởi con trỏ block.

Ví dụ: Ở ví dụ trên, sau khi thực hiện xong, thu hồi vùng nhớ cho 2 biến con trỏ pa  
và pb như sau:

```
free(pa); free(pb);
```

# Chương 6: Các cấu trúc dữ liệu khác

1. Kiểu cấu trúc(Structure): là kiểu dữ liệu bao gồm nhiều thành phần có kiểu khác nhau, mỗi thành phần được gọi là một trường (field)

a. Khai báo:

Cách 1:

```
struct <Tên cấu trúc>
{
    <Kiểu> <Trường 1> ;
    <Kiểu> <Trường 2> ;
    .....
    <Kiểu> <Trường n> ;
};
```

Cách 2: Sử dụng từ khóa typedef để định nghĩa kiểu:

```
typedef struct
{
    <Kiểu> <Trường 1> ;
    <Kiểu> <Trường 2> ;
    .....
    <Kiểu> <Trường n> ;
} <Tên cấu trúc>;
```

# 1. Kiểu cấu trúc(tiếp)

```
struct svien
{
    char hoten[30];
    char diachi[40];
    float dtb;
};
```

```
typedef struct
{
    char hoten[30];
    char diachi[40];
    float dtb;
} sinhvien;
```

Khai báo biến:

```
struct svien sv1,sv2;
sinhvien sv3, sv4;
```

b. Truy xuất đến một trường của biến cấu trúc

Cách 1: <biến cấu trúc>.<tên trường>

Cách 2: <biến con trỏ đến cấu trúc> -> <tên trường>

# Ví dụ về kiểu cấu trúc

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
typedef struct
{ unsigned char Ngay,Thang,Nam;
} NgayThang;

typedef struct
{ char MSSV[10];
char HoTen[40];
NgayThang NgaySinh;
int Phai;
char DiaChi[40];
} SinhVien;

/* Hàm in lên màn hình 1 mẫu tin SinhVien*/
void InSV(SinhVien s)
{
    printf("%7s %20s %2d-%2d-%4d %s\n",s.MSSV,s.HoTen,
s.NgaySinh.Ngay,s.NgaySinh.Thang,s.NgaySinh.Nam,s.DiaChi);
}
```

# Ví dụ về kiểu cấu trúc(tiếp)

```
int main()
{
    SinhVien SV, s;
    printf("Nhập MSSV: "); gets(SV.MSSV);
    printf("Nhập Họ và tên: "); gets(SV.HoTen);
    printf("Sinh ngày: "); scanf("%d", &SV.NgaySinh.Ngay);
    printf("Tháng: "); scanf("%d", &SV.NgaySinh.Thang);
    printf("Nam: "); scanf("%d", &SV.NgaySinh.Nam);
    printf("Giới tính (0: Nữ), (1: Nam): "); scanf("%d", &SV.Phai);
    flushall();
    printf("Địa chỉ: "); gets(SV.DiaChi);
    InSV(SV);
    s=SV; /* Gán trị cho mẫu tin s*/
    InSV(s);
    getch();
    return 0;
}
```

# Ví dụ về con trỏ đến kiểu cấu trúc

```
#include<conio.h>
#include<stdio.h>
typedef struct
{ unsigned char Ngay,Thang,Nam;
} NgayThang;
int main()
{
    NgayThang Ng={25,2,2007};
    NgayThang *p;
    p=&Ng;
    printf("Truy cap cau truc thong thuong %d-%d-%d\n",
          Ng.Ngay,Ng.Thang,Ng.Nam);
    printf("Truy cap qua con tro %d-%d-%d\n",
          p->Ngay,p->Thang,p->Nam);
    printf("Truy cap qua vung nho con tro %d-%d-%d\n",
          (*p).Ngay,(*p).Thang,(*p).Nam);
    getch();
    return 0;
}
```

## 2. Kiểu tập tin

### a. Khái niệm:

- Là kiểu dữ liệu cho phép lưu trữ dữ liệu ở bộ nhớ ngoài (đĩa). Khi kết thúc chương trình thì dữ liệu vẫn còn do đó chúng ta có thể sử dụng nhiều lần.
- Ngoài ra, kiểu tập tin có thể có kích thước lớn với số lượng các phần tử không hạn chế (chỉ bị hạn chế bởi dung lượng của bộ nhớ ngoài).
- **Biến tập tin:** là một biến thuộc kiểu dữ liệu tập tin dùng để đại diện cho một tập tin. Dữ liệu chứa trong một tập tin được truy xuất qua các thao tác với thông số là biến tập tin đại diện cho tập tin đó.
- **Con trỏ tập tin:** Khi một tập tin được mở ra để làm việc, tại mỗi thời điểm, sẽ có một vị trí của tập tin mà tại đó việc đọc/ghi thông tin sẽ xảy ra. Người ta hình dung có một con trỏ đang chỉ đến vị trí đó và đặt tên nó là con trỏ tập tin.

### b. Khai báo:

**FILE <Đanh sách các biến con trỏ> ;**

Ví dụ:            FILE \*f1,\*f2;

## c. Các thao tác trên tập tin

+Mở tập tin

Cú pháp: ***FILE \*fopen(char \*Path, const char \*Mode)***

Trong đó:

- Path: chuỗi chỉ đường dẫn đến tập tin trên đĩa.
- Type: chuỗi xác định cách thức mà tập tin sẽ mở.

Ví dụ: Mở một tập tin tên C:\BAITAP\VIDU.TXT để ghi.

```
FILE *f;  
f = fopen("C:\\BAITAP\\VIDU.TX", "w");  
if (f!=NULL)  
{  
    /* Các câu lệnh để thao tác với tập tin*/  
  
    /* Đóng tập tin*/  
}
```

# Bảng các giá trị của Mode

Chế độ	Ý nghĩa
r	Mở tập tin văn bản để đọc
w	Tạo ra tập tin văn bản mới để ghi
a	Nối vào tập tin văn bản
rb	Mở tập tin nhị phân để đọc
wb	Tạo ra tập tin nhị phân để ghi
ab	Nối vào tập tin nhị phân
r+	Mở một tập tin văn bản để đọc/ghi
w+	Tạo ra tập tin văn bản để đọc/ghi
a+	Nối vào hay tạo mới tập tin văn bản để đọc/ghi
r+b	Mở ra tập tin nhị phân để đọc/ghi
w+b	Tạo ra tập tin nhị phân để đọc/ghi
a+b	Nối vào hay tạo mới tập tin nhị phân

## c. Các thao tác trên tập tin

- + Hàm đóng tập tin: Hàm fclose() được dùng để đóng tập tin được mở bởi hàm fopen(). Hàm này sẽ ghi dữ liệu còn lại trong vùng đệm vào tập tin và đóng lại tập tin.

Cú pháp: *int fclose(FILE \*f)*

- + Hàm kiểm tra đến cuối tập tin hay chưa?

Cú pháp: *int feof(FILE \*f)*

Kiểm tra xem đã chạm tới cuối tập tin hay chưa và trả về EOF nếu cuối tập tin được chạm tới, ngược lại trả về 0.

- + Hàm di chuyển con trỏ tập tin về đầu tập tin - Hàm rewind()

Khi ta đang thao tác một tập tin đang mở, con trỏ tập tin luôn di chuyển về phía cuối tập tin. Muốn cho con trỏ quay về đầu tập tin như khi mở nó, ta sử dụng hàm rewind().

Cú pháp: *void rewind(FILE \*f)*

# d. Truy cập vào tập tin văn bản

- + Hàm ghi dữ liệu lên tập tin văn bản: Hàm putc()

Hàm này được dùng để ghi một ký tự lên một tập tin văn bản đang được mở để làm việc.

Cú pháp: *int putc(int c, FILE \*f)*

Trong đó, tham số c chứa mã Ascii của một ký tự nào đó. Mã này được ghi lên tập tin liên kết với con trỏ f. Hàm này trả về EOF nếu gặp lỗi.

- + Hàm này dùng để ghi một chuỗi ký tự chứa trong vùng đệm lên tập tin văn bản.

Cú pháp: *int puts(const char \*buffer, FILE \*f)*

Trong đó, buffer là con trỏ có kiểu char chỉ đến vị trí đầu tiên của chuỗi ký tự được ghi vào. Hàm này trả về giá trị 0 nếu buffer chứa chuỗi rỗng và trả về EOF nếu gặp lỗi.

Ví dụ: Viết chương trình ghi chuỗi ký tự lên tập tin văn bản C:\BAITHO.TXT

```
#include<stdio.h>
#include<conio.h>
int main()
{
    FILE *f;
    f=fopen("C:\\BAITHO","r+");
    if (f!=NULL) {
        fputs("Chi co thuyen moi hieu.\n",f);
        fputs("Bien menh mong duong nao.",f);
        fclose(f);
    }
    getch();
    return 0;
}
```

## d. Truy cập vào tập tin văn bản(tiếp)

- + Hàm dùng để ghi dữ liệu có định dạng lên tập tin văn bản.

Cú pháp: *fprintf(FILE \*f, const char \*format, varexpr)*

Trong đó: format: chuỗi định dạng (giống với các định dạng của hàm printf()), varexpr: danh sách các biểu thức, mỗi biểu thức cách nhau dấu phẩy (,).

- + Hàm dùng để đọc dữ liệu từ tập tin văn bản đang được mở để làm việc.

Cú pháp: *int getc(FILE \*f)*

Hàm này trả về mã Ascii của một ký tự nào đó (kể cả EOF) trong tập tin liên kết với con trỏ f.

- + Hàm đọc dữ liệu từ tập tin văn bản vào danh sách các biến theo định dạng.

Cú pháp: *fscanf(FILE \*f, const char \*format, varlist)*

Trong đó: format: chuỗi định dạng (giống hàm scanf()); varlist: danh sách các biến mỗi biến cách nhau dấu phẩy (,).

# Ví dụ

Viết chương trình chép tập tin C:\BAITHO.TXT ở trên sang tập tin D:\BAIHAT.TXT.

```
#include<stdio.h>
#include<conio.h>
int main()
{
    FILE *f1,*f2;
    f1=fopen("C:\\BAITHO.TXT","rt");
    f2=fopen("D:\\BAIHAT.TXT","wt");
    if (f1!=NULL && f2!=NULL)
    {
        int ch=fgetc(f1);
        while (!feof(f1))
        {
            fputc(ch,f2);
            ch=fgetc(f1);
        }
        fcloseall();
    }
    getch();
    return 0;
}
```

## e. Truy cập vào tập tin nhị phân

+ Hàm ghi dữ liệu lên tập tin nhị phân - Hàm fwrite()

**Cú pháp:** `size_t fwrite(const void *ptr, size_t size, size_t n, FILE *f)`

Trong đó:

- ptr: con trỏ chỉ đến vùng nhớ chứa thông tin cần ghi lên tập tin.
- n: số phần tử sẽ ghi lên tập tin.
- size: kích thước của mỗi phần tử.
- f: con trỏ tập tin đã được mở.
- Giá trị trả về của hàm này là số phần tử được ghi lên tập tin. Giá trị này bằng n trừ khi xuất hiện lỗi.

+ Đọc dữ liệu từ tập tin nhị phân - Hàm fread()

**Cú pháp:** `size_t fread(const void *ptr, size_t size, size_t n, FILE *f)`

Trong đó:

- ptr: con trỏ chỉ đến vùng nhớ sẽ nhận dữ liệu từ tập tin.
- n: số phần tử được đọc từ tập tin.
- size: kích thước của mỗi phần tử.
- f: con trỏ tập tin đã được mở.
- Giá trị trả về của hàm này là số phần tử đã đọc được từ tập tin. Giá trị này bằng n hay nhỏ hơn n nếu đã chạm đến cuối tập tin hoặc có lỗi xuất hiện..

## e. Truy cập vào tập tin nhị phân(tiếp)

### + Hàm di chuyển con trỏ tập tin - Hàm fseek()

- Việc ghi hay đọc dữ liệu từ tập tin sẽ làm cho con trỏ tập tin dịch chuyển một số byte, đây chính là kích thước của kiểu dữ liệu của mỗi phần tử của tập tin.
- Khi đóng tập tin rồi mở lại nó, con trỏ luôn ở vị trí ngay đầu tập tin. Nhưng nếu ta sử dụng kiểu mở tập tin là “a” để ghi nối dữ liệu, con trỏ tập tin sẽ di chuyển đến vị trí cuối cùng của tập tin này.
- Ta cũng có thể điều khiển việc di chuyển con trỏ tập tin đến vị trí chỉ định bằng hàm fseek().

**Cú pháp: int fseek(FILE \*f, long offset, int whence)**

Trong đó:

- f: con trỏ tập tin đang thao tác.
- offset: số byte cần dịch chuyển con trỏ tập tin kể từ vị trí trước đó. Phần tử đầu tiên là vị trí 0.
- whence: vị trí bắt đầu để tính offset, ta có thể chọn điểm xuất phát là:
  - 0 SEEK\_SET Vị trí đầu tập tin
  - 1 SEEK\_CUR Vị trí hiện tại của con trỏ tập tin
  - 2 SEEK\_END Vị trí cuối tập tin
- Kết quả trả về của hàm là 0 nếu việc di chuyển thành công. Nếu không thành công, 1 giá trị khác 0 (đó là 1 mã lỗi) được trả về.

# Phụ lục: Sự khác nhau giữa ngôn ngữ lập trình C và C++

# 7.1 Các ví dụ về C++

- Chú thích - comment

```
// A first program in C++.
```

- Làm tài liệu cho các chương trình
- Làm chương trình dễ đọc dễ hiểu hơn
- được trình biên dịch (compiler) bỏ qua
- 1 dòng chú thích bắt đầu với //

- Các định hướng tiền xử lý - directive

```
#include <iostream>
```

- Được xử lý ngay trước khi biên dịch
- Bắt đầu bằng #

hàm **main** trả về một giá trị kiểu số nguyên.

```
1 /* A first program in C++.  
2 Print a line of text to standard output */
```

Chú thích

```
3 #include <iostream>
```

```
4  
5 // function main begins program
```

Định hướng tiền xử lý (preprocessor directive) để khai báo sử dụng thư viện ra/vào chuẩn **<iostream>**.

```
6 int main()
```

```
7 {
```

```
8     std::cout << "Hello World!\n";
```

Viết một dòng ra output chuẩn (màn hình)

```
9  
10    return 0; // indicates successful completion
```

Ngoặc trái { bắt đầu thanh nam.

hàm **main** xuất hiện đúng một lần trong mỗi chương trình C++.

```
11  
12 } // end function main
```

Các lệnh kết thúc bằng dấu chấm phẩy ;

10  
11  
12

Tương ứng, ngoặc phải } kết thúc thân hàm.

Từ khóa **return** là một cách thoát khỏi hàm; giá trị 0 được trả về có nghĩa chương trình kết thúc thành công.

# Ví dụ 1b

```
● 1 // Printing a line with multiple statements.  
● 2 #include <iostream>  
● 3  
● 4 // function main begins program execution  
● 5 int main()  
● 6 {  
● 7     std::cout << "Welcome ";  
● 8     std::cout << "to C++!\n";  
● 9  
● 10    return 0; // indicate that program ended successfully  
● 11  
● 12 } // end function main
```

Nhiều dòng lệnh tạo output  
trên một dòng.

Welcome to C++!

# Ví dụ 1c

- 1 // Printing multiple lines with a single statement
- 2 #include <iostream>
- 3
- 4 // function main begins program execution
- 5 int main()
- 6 {
- 7 std::cout << "Welcome\ninto\n\nC++!\n";
- 8
- 9 return 0; // indicate that program ended successfully
- 10
- 11 } // end function main

Dùng ký tự dòng mới \n để in trên nhiều dòng.

Welcome  
to  
C++!

## Ví dụ 2: Chương trình tính tổng hai số nguyên

```
• 1 // Addition program.  
• 2 #include <iostream>  
• 3  
• 4 // function main begins program execution  
• 5 int main()  
{  
• 6     int integer1; // first number to be input by user  
• 7     int integer2; // second number to be input by user  
• 8     int sum; // variable in which sum will be stored  
• 9  
• 10    std::cout << "Enter first integer\n"; // prompt  
• 11    std::cin >> integer1; // read first integer  
• 12  
• 13    std::cout << "Enter second integer\n";  
• 14    std::cin >> integer2; // read second integer  
• 15  
• 16    sum = integer1 + integer2; // assign result to sum  
• 17  
• 18    std::cout << "Sum is " << sum << endl; // print sum  
• 19  
• 20  
• 21    return 0; // indicate that program ended successfully  
• 22  
• 23 } // end function main
```

Khai báo các biến nguyên.

Nhập một số nguyên từ input chuẩn, ghi vào biến integer1. `endl` cho kết quả là một dòng trống.

Tính toán có thể được thực hiện trong lệnh output: Thay cho các dòng 17 và 19:

`cout << "Sum is " << integer1 + integer2 << endl;`

Enter first integer  
45  
Enter second integer  
72  
Sum is 117

## 7.2 Biến chương trình

```
std::cin >> integer1;
```

- giả sử người dùng nhập 45

integer1 45

```
std::cin >> integer2;
```

- giả sử người dùng nhập 72

integer1 45  
integer2 72

```
sum = integer1 + integer2;
```

integer1 45  
integer2 72  
sum 117

## 7.3 Vào ra dữ liệu

Các đối tượng vào/ra cơ bản

- **cin**
  - dòng dữ liệu vào chuẩn - Standard input stream
  - thường là từ bàn phím
- **cout**
  - dòng dữ liệu ra chuẩn - Standard output stream
  - thường là màn hình máy tính
- **cerr**
  - dòng báo lỗi chuẩn - Standard error stream
  - hiện các thông báo lỗi

## 7.3 Vào ra dữ liệu (tiếp)

In dòng văn bản ra màn hình

```
std::cout << "Enter first integer\n"; // prompt
```

- Đối tượng ra chuẩn - Standard output stream object
  - **std::cout**
  - “nối” với màn hình
  - **<<**
    - toán tử chèn vào dòng dữ liệu ra – stream insert operator
    - giá trị bên phải (right operand) được chèn vào dòng dữ liệu ra
- Không gian tên - Namespace
  - **std::** có nghĩa là sử dụng tên thuộc “namespace” **std**
  - **std::** được bỏ qua nếu dùng các khai báo **using**
- Escape characters \
  - đánh dấu các ký tự đặc biệt
    - ví dụ \\, \', \n, \t

## 7.3 Vào ra dữ liệu

### Các chuỗi escape

Chuỗi Escape	Mô tả
\n	Dòng mới. Đ . . . . . àn . . . . . đ . . . . . òng . . . . .
\t	Tab. Di chuyển con trỏ đ . . . . .
\r	Về đ . . . . . òng . . . . . đ . . . . . òng . . . . .
\a	Chuông. Bật chuông hệ thống.
\\"	Chéo ngược . . . . .
\ "	Nháy kép. Dùng đ . . . . .

## 7.3 Vào ra dữ liệu

Nhập dữ liệu từ thiết bị vào chuẩn

```
std::cin >> integer1; // read an integer
```

- Đối tượng dòng dữ liệu vào - Input stream object
  - `>>` (toán tử đọc từ dòng dữ liệu vào)
    - được sử dụng với `std::cin`
    - đợi người dùng nhập giá trị, rồi gõ phím *Enter* (Return)
    - lưu giá trị vào biến ở bên phải toán tử
      - đổi giá trị được nhập sang kiểu dữ liệu của biến
- `=` (toán tử gán)
  - gán giá trị cho biến
  - toán tử hai ngôi - Binary operator
  - Ví dụ:

```
sum = variable1 + variable2;
```

# Ví dụ khai báo biến điều kiện

- ```
1 // Using if statements, relational
2 // operators, and equality operators.
3 #include <iostream>
4
5 using std::cout; // program uses cout
6 using std::cin; // program uses cin
7 using std::endl; // program uses endl
8
9 // function main begins program execution
10 int main()
11 {
12     int num1; // first number to be read from user
13     int num2; // second number to be read from user
14
15     cout << "Enter two integers, and I will tell you "
16         << "the relationships they satisfy: ";
17     cin >> num1 >> num2; // read two integers
18
19     if ( num1 == num2 )
20         cout << num1 << " is equal to " << num2 << endl;
21
22     if ( num1 != num2 )
23         cout << num1 << " is not equal to " << num2 << endl;
```

Khai

## Khai báo biển.

i báo biến. **Đảo ẩn** là cách để sau đó  
cần dùng tiền tố **std::**:

lệnh **if** kiểm tra xem các giá trị của **num1** và **num2** có bằng nhau

Nếu điều kiện là đúng (nghĩa là hai giá trị bằng nhau) thì

lệnh **if** kiểm tra xem các giá trị của **num1** và **num2** có khác nhau không.

Nếu điều kiện là đúng (nghĩa là hai giá trị khác nhau) thì thực hiện lệnh này.

# Ví dụ khai báo biến điều kiện

```
• 24 if ( num1 < num2 )
• 25     cout << num1 << " is less than " << num2 << endl;
• 26
• 27 if ( num1 > num2 )
• 28     cout << num1 << " is greater than " << num2 << endl;
• 29
• 30 if ( num1 <= num2 )
• 31     cout << num1 << " is less than or equal to "
• 32         << num2 << endl;
• 33
• 34 if ( num1 >= num2 )
• 35     cout << num1 << " is greater than or equal to "
• 36         << num2 << endl;
• 37
• 38 return 0; // indicate that program ended successfully
• 39
• 40 } // end function main
```

Một lệnh có thể được tách thành nhiều dòng.

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

# Ví dụ khai báo biến điều kiện

Enter two integers, and I will tell you  
the relationships they satisfy: 7 7

7 is equal to 7

7 is less than or equal to 7

7 is greater than or equal to 7

## 7.4 Khai báo using

- Khai báo sử dụng toàn bộ không gian tên
  - using namespace std;
  - Để không cần tiền tố std:: cho mọi tên trong std

```
1 // Printing a line with multiple statements.  
2 #include <iostream>  
3  
4 using namespace std;  
5 // function main begins program execution  
6 int main()  
7 {  
8     cout << "Welcome "  
9     std::cout << "to C++!\n";  
10  
11    return 0;  
12  
13 } // end function main
```

## 7.4 Khai báo using

- Khai báo sử dụng cùng tên

```
using std::cout; // program uses cout
using std::cin; // program uses cin
using std::endl; // program uses endl
...
cout << "No need to write std::";
cin >> somevariable;
...
```

# KỸ THUẬT LẬP TRÌNH

---



LỚP KOMTUM 2015  
GIÁNG VIÊN: NGUYỄN ĐÌNH CƯỜNG

# KỸ THUẬT QUY HOẠCH ĐỘNG

BỨC TƯỜNG ĐƯỢC XÂY LÊN TỪ NHỮNG VIÊN GẠCH NHỎ, TỪ NHỮNG BỨC TƯỜNG TÀ CÓ ĐƯỢC NGÔI NHÀ

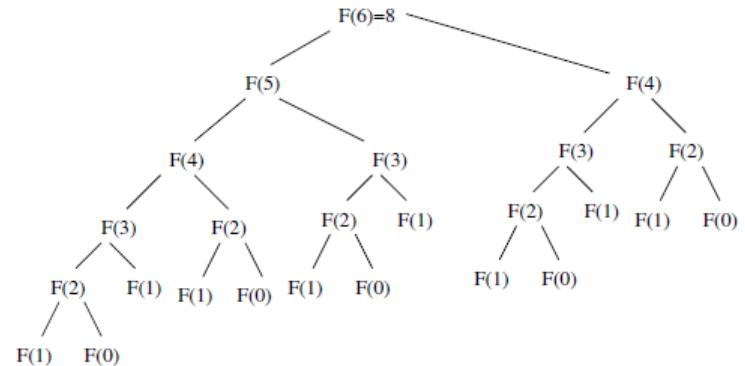
Bài toán số fibonaci nhìn lại

$$F_n = F_{n-1} + F_{n-2}$$

$F_0 = 0$  and  $F_1 = 1$ . Thus,  $F_2 = 1$ ,  $F_3 = 2$

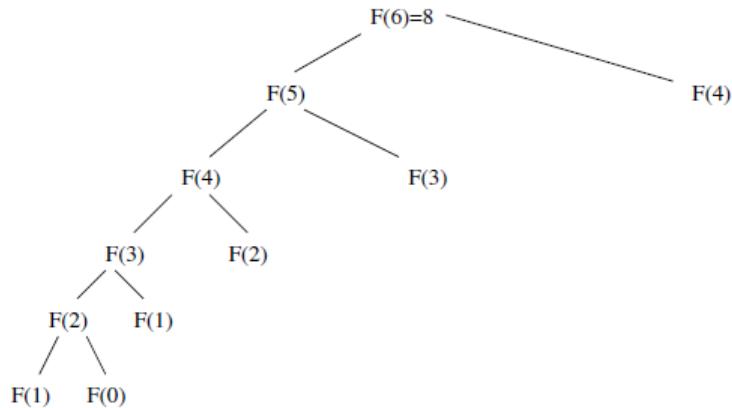
```
long fib_r(int n)
{
    if (n == 0) return(0);
    if (n == 1) return(1);

    return(fib_r(n-1) + fib_r(n-2));
}
```



```
#define MAXN    45      /* largest interesting n */
#define UNKNOWN -1       /* contents denote an empty cell */
long f[MAXN+1];           /* array for caching computed fib values */
```

# KỸ THUẬT QUY HOẠCH ĐỘNG



```
long fib_dp(int n)
{
    int i; /* counter */
    long f[MAXN+1]; /* array to cache computed fib values */

    f[0] = 0;
    f[1] = 1;
    for (i=2; i<=n; i++) f[i] = f[i-1]+f[i-2];

    return(f[n]);
}
```

```
long fib_ultimate(int n)
{
    int i; /* counter */
    long back2=0, back1=1; /* last two values of f[n] */
    long next; /* placeholder for sum */

    if (n == 0) return (0);

    for (i=2; i<n; i++) {
        next = back1+back2;
        back2 = back1;
        back1 = next;
    }
    return(back1+back2);
}

long fib_c(int n)
{
    if (f[n] == UNKNOWN)
        f[n] = fib_c(n-1) + fib_c(n-2);

    return(f[n]);
}

long fib_c_driver(int n)
{
    int i; /* counter */
    f[0] = 0;
    f[1] = 1;
    for (i=2; i<=n; i++) f[i] = UNKNOWN;

    return(fib_c(n));
}
```

# KỸ THUẬT QUY HOẠCH ĐỘNG

## Bài toán độ sai khác của 2 chuỗi

$D[i, j]$  : là khoảng cách ngắn nhất của chuỗi S độ dài i, so với chuỗi T độ dài j. Ta có 3 khả năng sau:

1.  $D[i, j] = D[i - 1, j - 1]$  Nếu  $S_i = T_j$

Ngược lại  $D[i, j] = D[i - 1, j - 1] + 1$

2.  $D[i, j] = D[i - 1, j] + 1$  cho trường hợp **chèn** một kí tự vào chuỗi S tại vị trí i.

3.  $D[i, j] = D[i, j - 1] + 1$  cho trường hợp **xóa** một kí tự ở chuỗi P tại vị trí j.

## Lời giải đệ quy

```
#define MATCH      0      /* enumerated type symbol for match */
#define INSERT     1      /* enumerated type symbol for insert */
#define DELETE     2      /* enumerated type symbol for delete */
```

```
int match(char c, char d)           int indel(char c)
{
    if (c == d) return(0);
    else return(1);
}
```

```
int string_compare(char *s, char *t, int i, int j)
{
    int k;                      /* counter */
    int opt[3];                  /* cost of the three options */
    int lowest_cost;             /* lowest cost */

    if (i == 0) return(j * indel(' '));
    if (j == 0) return(i * indel(' '));

    opt[MATCH] = string_compare(s,t,i-1,j-1) + match(s[i],t[j]);
    opt[INSERT] = string_compare(s,t,i,j-1) + indel(t[j]);
    opt[DELETE] = string_compare(s,t,i-1,j) + indel(s[i]);

    lowest_cost = opt[MATCH];
    for (k=INSERT; k<=DELETE; k++)
        if (opt[k] < lowest_cost) lowest_cost = opt[k];

    return( lowest_cost );
}
```

# KỸ THUẬT QUY HOẠCH ĐỘNG

## Lời giải khử đê quy bằng quy hoạch động

```
typedef struct {
    int cost;           /* cost of reaching this cell */
    int parent;         /* parent cell */
} cell;

cell m[MAXLEN+1] [MAXLEN+1];      /* dynamic programming table */

int string_compare(char *s, char *t)
{
    int i,j,k;          /* counters */
    int opt[3];          /* cost of the three options */

    for (i=0; i<MAXLEN; i++) {
        row_init(i);
        column_init(i);
    }

    for (i=1; i<strlen(s); i++) {
        for (j=1; j<strlen(t); j++) {
            opt[MATCH] = m[i-1][j-1].cost + match(s[i],t[j]);
            opt[INSERT] = m[i][j-1].cost + indel(t[j]);
            opt[DELETE] = m[i-1][j].cost + indel(s[i]);
        }
    }
}
```

```
m[i][j].cost = opt[MATCH];
m[i][j].parent = MATCH;
for (k=INSERT; k<=DELETE; k++)
    if (opt[k] < m[i][j].cost) {
        m[i][j].cost = opt[k];
        m[i][j].parent = k;
    }
}
goal_cell(s,t,&i,&j);
return( m[i][j].cost );
}

goal_cell(char *s, char *t, int *i, int *j)
{
    *i = strlen(s) - 1;
    *j = strlen(t) - 1;
}
```

## KỸ THUẬT QUY HOẠCH ĐỘNG

Một ví dụ tính toán thông qua bảng dữ liệu của 2 chuỗi P và T

| P  | T   | 0        | y        | o        | u        | -        | s        | h        | o        | u        | l        | d        | -        | n        | o        | t        |
|----|-----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
|    | pos | 0        | 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        | 10       | 11       | 12       | 13       | 14       |
| :  |     | <b>0</b> | 1        | <b>2</b> | 3        | 4        | 5        | 6        | 7        | 8        | 9        | 10       | 11       | 12       | 13       | 14       |
| t: | 1   | <b>1</b> | 1        | <b>2</b> | 3        | 4        | 5        | 6        | 7        | 8        | 9        | 10       | 11       | 12       | 13       | 13       |
| h: | 2   | 2        | <b>2</b> | <b>2</b> | 3        | 4        | 5        | 5        | 6        | 7        | 8        | 9        | 10       | 11       | 12       | 13       |
| o: | 3   | 3        | 3        | <b>2</b> | 3        | <b>4</b> | <b>5</b> | 6        | 5        | 6        | 7        | 8        | 9        | 10       | 11       | 12       |
| u: | 4   | 4        | 4        | 3        | <b>2</b> | <b>3</b> | <b>4</b> | 5        | 6        | 5        | 6        | 7        | 8        | 9        | 10       | 11       |
| -: | 5   | 5        | 5        | 4        | 3        | <b>2</b> | 3        | 4        | 5        | 6        | 6        | 7        | 7        | 8        | 9        | 10       |
| s: | 6   | 6        | 6        | 5        | 4        | 3        | <b>2</b> | 3        | 4        | 5        | 6        | 7        | 8        | 8        | 9        | 10       |
| h: | 7   | 7        | 7        | 6        | 5        | 4        | 3        | <b>2</b> | <b>3</b> | 4        | 5        | 6        | 7        | 8        | 9        | 10       |
| a: | 8   | 8        | 8        | 7        | 6        | 5        | 4        | 3        | 3        | <b>4</b> | 5        | 6        | 7        | 8        | 9        | 10       |
| l: | 9   | 9        | 9        | 8        | 7        | 6        | 5        | 4        | 4        | 4        | <b>4</b> | 5        | 6        | 7        | 8        | 9        |
| t: | 10  | 10       | 10       | 9        | 8        | 7        | 6        | 5        | 5        | 5        | 5        | <b>5</b> | 6        | 7        | 8        | 8        |
| -: | 11  | 11       | 11       | 10       | 9        | 8        | 7        | 6        | 6        | 6        | 6        | 6        | <b>5</b> | 6        | 7        | 8        |
| n: | 12  | 12       | 12       | 11       | 10       | 9        | 8        | 7        | 7        | 7        | 7        | 7        | 6        | <b>5</b> | 6        | 7        |
| o: | 13  | 13       | 13       | 12       | 11       | 10       | 9        | 8        | 7        | 8        | 8        | 8        | 7        | 6        | <b>5</b> | 6        |
| t: | 14  | 14       | 14       | 13       | 12       | 11       | 10       | 9        | 8        | 8        | 9        | 9        | 8        | 7        | 6        | <b>5</b> |

## KỸ THUẬT QUY HOẠCH ĐỘNG

### Ngăn sách

Bài toán chia sách cho mỗi người để cực tiểu hóa giá trị lớn nhất số trang sách cần chia, trong những người được đề nghị.

100 100 100 | 100 100 100 | 100 100 100

Trong trường hợp đặc biệt có 9 quyển sách mỗi quyển 100 trang, chia cho 3 người. Ta có phương án chia 300 trang cho mỗi người

Phát biểu bài toán tổng quát

Bài toán chia ngăn sách

Input: N cuốn sách mỗi cuốn sách có số trang tương ứng  $\{S_1, S_2, S_3, \dots, S_n\}$  và k ngăn cần chia.

Output: Phân số lượng sách vào k ngăn sao cho cực tiểu hóa số trang sách ở ngăn nhiều trang sách nhất, với thứ tự các quyển sách không thay đổi.

## KỸ THUẬT QUY HOẠCH ĐỘNG

### Ngăn sách, hướng dẫn lời giải

Gọi  $M[n, k]$  là khả năng cực tiểu có thể khi ngăn  $\{S_1, S_2, \dots, S_n\}$  thành  $k$  phần thỏa mãn điều kiện bài toán.

Từ yêu cầu ta có:

$$M[n, k] = \min_{i=1}^n \max(M[i, k-1], \sum_{j=i+1}^n s_j)$$

$$M[1, k] = s_1, \text{ for all } k > 0 \text{ and,}$$

$$M[n, 1] = \sum_{i=1}^n s_i$$

# KỸ THUẬT QUY HOẠCH ĐỘNG

```
partition(int s[], int n, int k)
{
    int m[MAXN+1][MAXK+1];
    int d[MAXN+1][MAXK+1];
    int p[MAXN+1];
    int cost;
    int i,j,x;

    p[0] = 0;                                /* construct prefix sums */
    for (i=1; i<=n; i++) p[i]=p[i-1]+s[i];

    for (i=1; i<=n; i++) m[i][1] = p[i];      /* initialize boundaries */
    for (j=1; j<=k; j++) m[1][j] = s[1];

    for (i=2; i<=n; i++)                      /* evaluate main recurrence */
        for (j=2; j<=k; j++) {
            m[i][j] = MAXINT;
            for (x=1; x<=(i-1); x++) {
                cost = max(m[x][j-1], p[i]-p[x]);
                if (m[i][j] > cost) {
                    m[i][j] = cost;
                    d[i][j] = x;
                }
            }
        }

    reconstruct_partition(s,d,n,k);           /* print book partition */
}
```

## Ngắn sách viết chương trình

```
reconstruct_partition(int s[],int d[MAXN+1][MAXK+1], int n, int k)
{
    if (k==1)
        print_books(s,1,n);
    else {
        reconstruct_partition(s,d,d[n][k],k-1);
        print_books(s,d[n][k]+1,n);
    }
}

print_books(int s[], int start, int end)
{
    int i;                               /* counter */

    for (i=start; i<=end; i++) printf(" %d ",s[i]);
    printf("\n");
}
```

## KỸ THUẬT QUY HOẠCH ĐỘNG

Ví dụ số

| <i>M</i> | <i>k</i> |  | <i>D</i> | <i>k</i> |
|----------|----------|--|----------|----------|
| <i>n</i> | 1 2 3    |  | <i>n</i> | 1 2 3    |
| 1        | 1 1 1    |  | 1        | — — —    |
| 1        | 2 1 1    |  | 1        | — 1 1    |
| 1        | 3 2 1    |  | 1        | — 1 2    |
| 1        | 4 2 2    |  | 1        | — 2 2    |
| 1        | 5 3 2    |  | 1        | — 2 3    |
| 1        | 6 3 2    |  | 1        | — 3 4    |
| 1        | 7 4 3    |  | 1        | — 3 4    |
| 1        | 8 4 3    |  | 1        | — 4 5    |
| 1        | 9 5 3    |  | 1        | — 4 6    |

Lưu vết ở cột ngăn

$\{1, 1, 1, 1, 1, 1, 1, 1, 1\}$  into  $\{\{1, 1, 1\}, \{1, 1, 1\}, \{1, 1, 1\}\}$

| <i>M</i> | <i>k</i> |  | <i>D</i> | <i>k</i> |
|----------|----------|--|----------|----------|
| <i>n</i> | 1 2 3    |  | <i>n</i> | 1 2 3    |
| 1        | 1 1 1    |  | 1        | — — —    |
| 2        | 3 2 2    |  | 2        | — 1 1    |
| 3        | 6 3 3    |  | 3        | — 2 2    |
| 4        | 10 6 4   |  | 4        | — 3 3    |
| 5        | 15 9 6   |  | 5        | — 3 4    |
| 6        | 21 11 9  |  | 6        | — 4 5    |
| 7        | 28 15 11 |  | 7        | — 5 6    |
| 8        | 36 21 15 |  | 8        | — 5 6    |
| 9        | 45 24 17 |  | 9        | — 6 7    |

$\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$\{\{1, 2, 3, 4, 5\}, \{6, 7\}, \{8, 9\}\}$

# KỸ THUẬT QUY HOẠCH ĐỘNG

## Tam giác pascal

|   |   |    |    |   |   |
|---|---|----|----|---|---|
|   | 1 |    |    |   |   |
|   | 1 | 1  |    |   |   |
| 1 | 2 | 1  |    |   |   |
| 1 | 3 | 3  | 1  |   |   |
| 1 | 4 | 6  | 4  | 1 |   |
| 1 | 5 | 10 | 10 | 5 | 1 |

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

## Chương trình

```
long binomial_coefficient(n,m)
int n,m;                                /* computer n choose m */
{
    int i,j;                            /* counters */
    long bc[MAXN][MAXN];      /* table of binomial coefficients */

    for (i=0; i<=n; i++) bc[i][0] = 1;
    for (j=0; j<=n; j++) bc[j][j] = 1;

    for (i=1; i<=n; i++)
        for (j=1; j<i; j++)
            bc[i][j] = bc[i-1][j-1] + bc[i-1][j];

    return( bc[n][m] );
}
```

## Tài liệu tham khảo

- [1]. Bài giảng lập trình C, Nguyễn Đình Thuân, Đại học Nha Trang, 2008.
- [2] Bài giảng Kỹ thuật lập trình, Nguyễn Ngọc Phương, Học viện bưu chính viễn thông.
- [3] Bài giảng toán rời rạc, Đỗ Như An, Đại học Nha Trang, 2008.
- [4] Lập trình C cơ bản( <http://tuhocanninhmang.com>, <http://www.cplusplus.com>).
- [5] The algorithm Design Manual, Steven S. Skiena, 2008
- [6] Lập trình C mô phỏng thế giới thật.
- [7] Numerical Recipes in C, the Art of Scientific Computing Second Edition

## KỸ THUẬT ĐA TIỀN TRÌNH

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

// Let us create a global variable to change it in threads
int g = 0;

// The function to be executed by all threads
void *myThreadFun(void *vargp)
{
    // Store the value argument passed to this thread
    int *myid = (int *)vargp;

    // Let us create a static variable to observe its changes
    static int s = 0;

    // Change static and global variables
    ++s; ++g;

    // Print the argument, static and global variables
    printf("Thread ID: %d, Static: %d, Global: %d\n", *myid, ++s, ++g);
}

int main()
{
    int i;
    pthread_t tid;

    // Let us create three threads
    for (i = 0; i < 3; i++)
        pthread_create(&tid, NULL, myThreadFun, (void *)i);

    pthread_exit(NULL);
    return 0;
}
```

```
gfg@ubuntu:~/ $ gcc multithread.c -lpthread
gfg@ubuntu:~/ $ ./a.out
Thread ID: 1, Static: 1, Global: 1
Thread ID: 0, Static: 2, Global: 2
Thread ID: 2, Static: 3, Global: 3
gfg@ubuntu:~/ $
```

## KỸ THUẬT ĐA TIỀN TRÌNH

```
#include <iostream>
#include <cstdlib>
#include <pthread.h>

using namespace std;

#define NUM_THREADS 5

void *PrintHello(void *threadid) {
    long tid;
    tid = (long)threadid;
    cout << "Hello World! Thread ID, " << tid << endl;
    pthread_exit(NULL);
}

int main () {
    pthread_t threads[NUM_THREADS];
    int rc;
    int i;

    for( i = 0; i < NUM_THREADS; i++ ) {
        cout << "main() : creating thread, " << i << endl;
        rc = pthread_create(&threads[i], NULL, PrintHello, (void *)i);

        if (rc) {
            cout << "Error:unable to create thread," << rc << endl;
            exit(-1);
        }
    }
    pthread_exit(NULL);
}
```

Compile the following program using -lpthread library as follows –

```
$gcc test.cpp -lpthread
```

Now, execute your program which gives the following output –

```
main() : creating thread, 0
main() : creating thread, 1
main() : creating thread, 2
main() : creating thread, 3
main() : creating thread, 4
Hello World! Thread ID, 0
Hello World! Thread ID, 1
Hello World! Thread ID, 2
Hello World! Thread ID, 3
Hello World! Thread ID, 4
```

# GÓI TIẾNG VIỆT UNICODE

The screenshot shows a Visual Studio Code interface with a dark theme. On the left is the code editor containing a file named 'xincho.cpp'. The code is a simple C++ program that sets up a console window to display Vietnamese text. On the right is a terminal window titled 'Việt Nam Vô Địch!' showing the output of the program. The terminal output includes a greeting in Vietnamese, a prompt for input, and a message indicating that the input was successfully read. The status bar at the bottom provides information about the file ('xincho.cpp'), encoding ('UTF-16 LE'), and other settings.

```
// xincho.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <iostream>
#include <io.h>
#include <fcntl.h>
#include <string>
#include <windows.h>
#include <fstream>

int main()
{
    _setmode(_fileno(stdin), _O_U16TEXT);
    _setmode(_fileno(stdout), _O_U16TEXT);
    SetConsoleTitleW(L"Việt Nam Vô Địch!");
    HANDLE hdlConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_FONT_INFOEX consoleFont;
    consoleFont.cbSize = sizeof(consoleFont);
    GetCurrentConsoleFontEx(hdlConsole, FALSE, &consoleFont);
    memcpy(consoleFont.FaceName, L"Consolas", sizeof(consoleFont.FaceName));
    SetCurrentConsoleFontEx(hdlConsole, FALSE, &consoleFont);

    std::wcout << L"Tiếng Việt có dấu" << std::endl;
    std::wstring test;
    std::wcout << L"Hãy nhập vào một chuỗi ký tự:" << std::endl;
    std::getline(std::wcin, test);
    std::wcout << L"Chuỗi ký tự mà bạn vừa mới nhập:" << std::endl;
    std::wcout << test << std::endl;

    FILE *f;
```

Việt Nam Vô Địch!

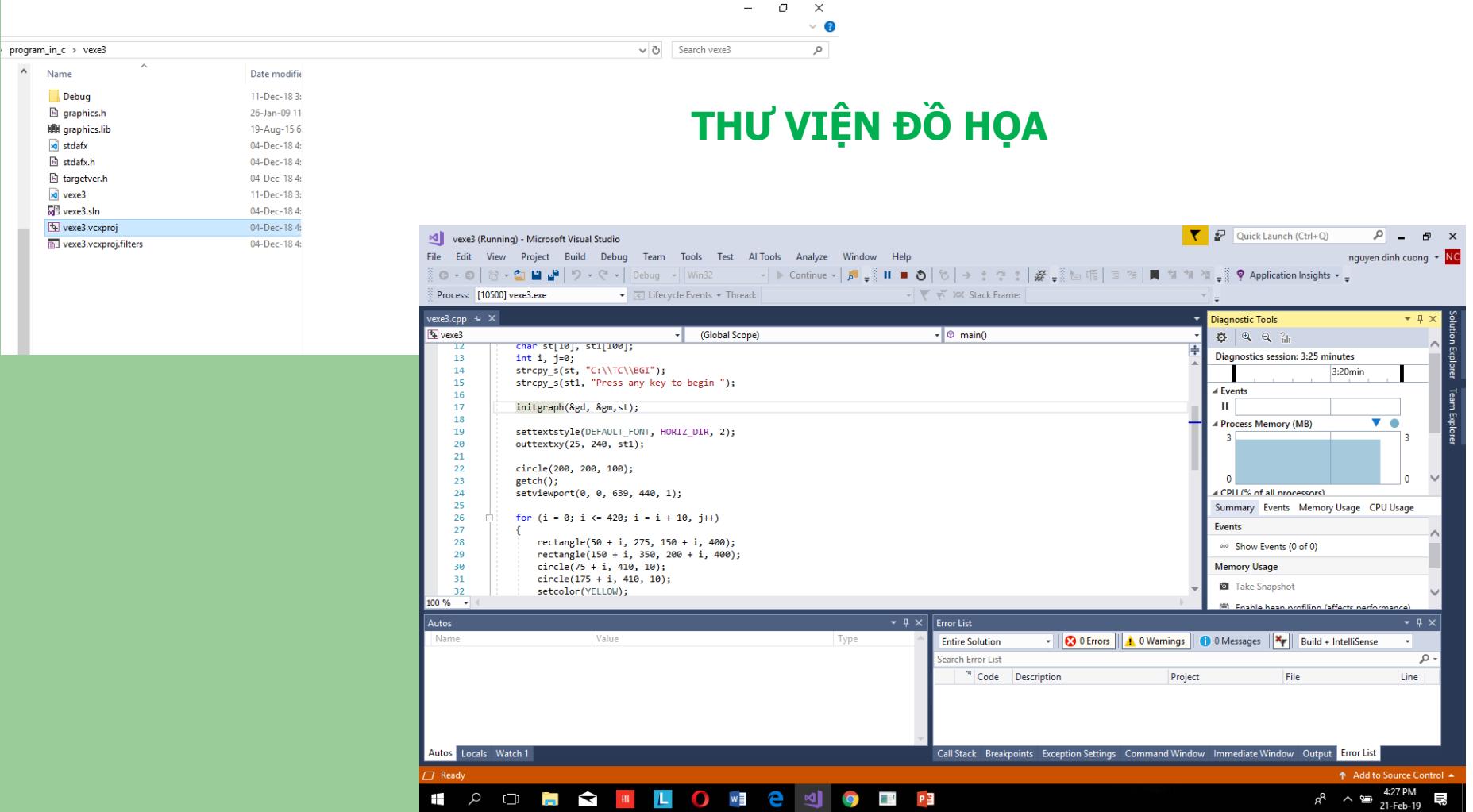
Tiếng Việt có dấu  
Hãy nhập vào một chuỗi ký tự:  
Xin chào các bạn sinh viên Việt Nam  
Chuỗi ký tự mà bạn vừa mới nhập:  
Xin chào các bạn sinh viên Việt Nam  
Press any key to continue . . .

Ln 1, Col 1 Tab Size: 4 UTF-16 LE CRLF C++ Win32 4:32 PM 21-Feb-19

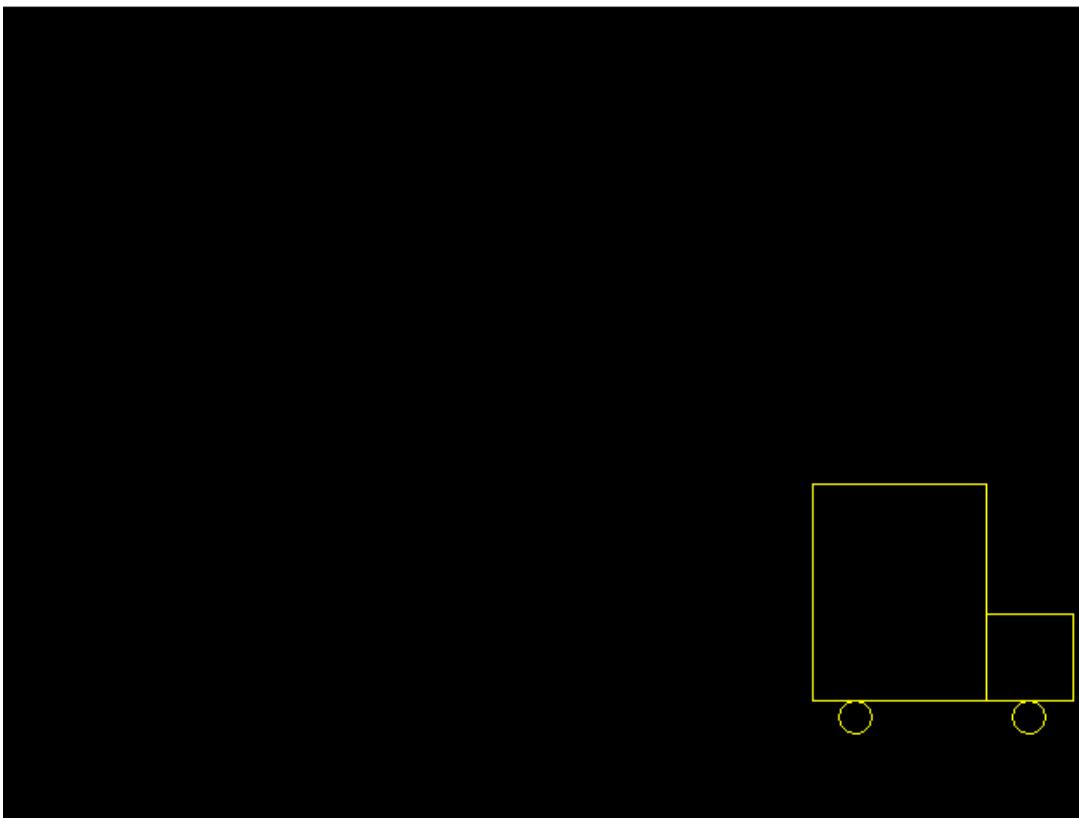
# BẢNG MÃ UNICODE

|      |   |   |    |    |   |    |   |    |   |    |   |    |   |   |   |    |
|------|---|---|----|----|---|----|---|----|---|----|---|----|---|---|---|----|
| 00AO | İ | € | £  | ¤  | ¥ |    | § | „  | © | ¤  | « | „  | ® | - |   |    |
| 00BO | ° | ± | ²  | ³  | ‘ | μ  | ¶ | .  | , | ·  | ¤ | »  | ¼ | ½ | ¾ | ݰ  |
| 00CO | À | Á | Â  | Ã  | Ä | Å  | Æ | Ç  | È | É  | Ê | Ë  | Ì | Í | Î | Ї  |
| 00DO | Đ | Ñ | Ò  | Ó  | Ô | Õ  | Ö | ×  | Ø | Ù  | Ú | Û  | Ü | Ý | Þ | ݢ  |
| 00E0 | à | á | â  | ã  | ä | å  | æ | ç  | è | é  | ê | ë  | ì | í | î | ї  |
| 00F0 | ð | ñ | ò  | ó  | ô | õ  | ö | ÷  | ø | ù  | ú | û  | ü | ý | þ | ݢ  |
| 0100 | Ā | ā | Ă  | ă  | Ā | ą  | Ć | ć  | Ĉ | ĉ  | Ć | ć  | Č | č | Đ | d' |
| 0110 | Đ | đ | Ē  | ē  | Ě | ě  | Ē | ē  | Ē | ē  | Ē | ě  | Ĝ | ĝ | ݢ | ݢ  |
| 0120 | Ğ | ğ | Ğ  | ğ  | Ĥ | ĥ  | Ҥ | ҥ  | Ӣ | ӣ  | Ӣ | ӣ  | Ӣ | ҝ | ڶ | ڶ  |
| 0130 | Î | î | IJ | ij | Ĵ | ŷ  | K | k  | K | í  | L | í  | L | l | L | լ  |
| 0140 | Ľ | ľ | Ľ  | ń  | Ń | n̄ | Ń | ń  | ń | n̄ | N | n̄ | Ō | ō | ܽ | ܽ  |
| 0150 | Ő | ő | Œ  | œ  | Ŕ | r̄ | R | r̄ | Ŕ | r̄ | S | s̄ | ܶ | ܶ | ܶ | ܶ  |

# THƯ VIỆN ĐỒ HỌA



Windows BGI



# KỸ THUẬT LẬP TRÌNH

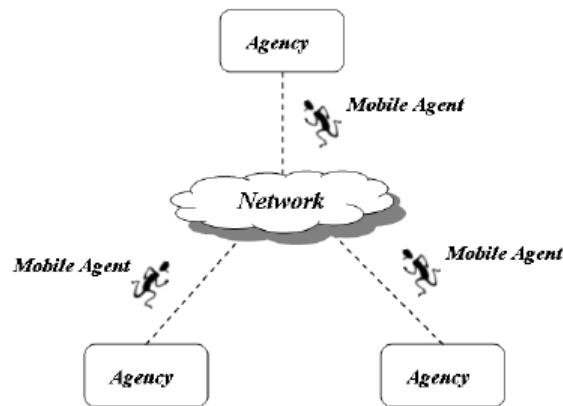
---



LỚP KONTUM 2015

GIẢNG VIÊN: NGUYỄN ĐÌNH CƯỜNG

# LẬP TRÌNH MOBILE



## Copyright

```
/*
 * Copyright (c) 2007-2008 Integration Engineering Laboratory
 * University of California, Davis
 *
 * Permission to use, copy, and distribute this software and its
 * documentation for any purpose with or without fee is hereby granted,
 * provided that the above copyright notice appear in all copies and
 * that both that copyright notice and this permission notice appear
 * in supporting documentation.
 *
 * Permission to modify the software is granted, but not the right to
 * distribute the complete modified source code. Modifications are to
 * be distributed as patches to the released version. Permission to
 * distribute binaries produced by compiling modified sources is granted,
 * provided you
 *   1. distribute the corresponding source modifications from the
 *      released version in the form of a patch file along with the binaries,
 *   2. add special version identification to distinguish your version
 *      in addition to the base release version number,
 *   3. provide your name and address as the primary contact for the
 *      support of your modified version, and
 *   4. retain our contact information in regard to use of the base
 *      software.
 * Permission to distribute the released version of the source code along
 * with corresponding source modifications in the form of a patch file is
 * granted with same provisions 2 through 4 for binary distributions.
 *
 * This software is provided "as is" without express or implied warranty
 * to the extent permitted by applicable law.
 */
```

## THE FINAL CLASS

# LẬP TRÌNH MOBILE

```
#include <libmc.h>

int main()
{
    MCAgency_t agency;
    MCAgencyOptions_t options;
    MCAgent_t agent;
    int local_port=5050;

    MC_InitializeAgencyOptions(&options);
    //MC_SetThreadOff(&options, MC_THREAD_CP); /* Turn off command prompt */
    agency = MC_Initialize(local_port, &options);

    agent = MC_ComposeAgentFromFile(
        "mobagent1", /* Name */
        "localhost:5050", /* Home */
        "IEL", /* Owner */
        "hello_world.c", /* Filename */
        NULL, /* Return var name. NULL for no return */
        "localhost:5051", /* Server to execute task on */
        0 ); /* Persistent. 0 for no persistence. */
```

(1)

```
/* File: hello_world/hello_world.c */
```

```
int main()
{
    printf("Hello world.\n");
    return 0;
}
```

(2.2)

```
/* Add the agent to the agency to start it */
MC_AddAgent(agency, agent);

MC_MainLoop(agency);
MC_End(agency);
exit(0);
}
```

(2.1)

SERVER program

# LẬP TRÌNH MOBILE

```
/* File: hello_world/server.c */

#include <stdio.h>
#include <libmc.h>

int main()
{
    MCAgency_t agency;

    int local_port = 5051;
    char embedchhome[] = "/home/dko/sys/ch7.0.0";
    ChOptions_t* ch_options;
    MCAgencyOptions_t mc_options;
    setbuf(stdout, NULL);

    printf("Initializing options...\n");
    ch_options = (ChOptions_t*)malloc(sizeof(ChOptions_t));
    MC_InitializeAgencyOptions(&mc_options);
    ch_options->shelltype = CH_REGULARCH;
    ch_options->chhome = malloc(strlen(embedchhome)+1);
    strcpy(ch_options->chhome, embedchhome);

    mc_options.ch_options = ch_options;

    printf("Initializing agency...\n");

    //agency = MC_Initialize(local_port, &mc_options);
    agency = MC_Initialize(local_port, NULL);
```

## SERVER PROGRAM

```
printf("Running mainloop... \n");
```

```
    MC_End(agency);
    return 0;
}
```

## AGENT DATA

- MESSAGE: This tag indicates to Mobile-C that the following data is a Mobile-C message. The message type is included in the attribute “message”.
- MOBILE\_AGENT: This tag indicates that the contained data is a Mobile-C agent.
- AGENT\_DATA: This tag indicates that the contained data is data pertaining to this particular agent.
- NAME: The name of the agent.
- OWNER: The owner of the agent.
- HOME: The home of the agent. Any agent that has data to “return” will return it to this address by default.
- WG\_CODE: The workgroup code of the agent. Workgroup codes are kept secret by the agent. Only agents with matching workgroup codes are allowed to perform certain operations on each other, such as agent deletion.
- TASKS: This indicates that the following information pertains to the task or tasks the agent is intended to perform. Attributes found under the TASKS tag include:
  - task : The total number of tasks the agent has.
  - num : The task that the agent is currently on.
- TASK: Each separate TASK tag indicates a separate task for the agent to perform. The tasks may be separate hosts and/or code blocks. In the rudimentary example, there is only one task. Listed below are the attributes of TASK tags.
  - num: The number of the task. The first task is task number zero.
  - complete: Completeness of the task
  - server: The host to perform the task

## LẬP TRÌNH MOBILE

will send an agent to the server at address “169.237.104.199” listening on port 5055. Or,

```
MC_ComposeAgentFromFile(
    "Bob",
    "iel.ucdavis.edu:5050",
    "IEL",
    "source_code.c",
    "169.237.104.199:5055",
    NULL,
    0);
```

```
MC_ComposeAgentFromFile(
    "Lou",
    "iel.ucdavis.edu:5055",
    "IEL",
    "agent_source.c",
    "machine.ucdavis.edu:5031",
    NULL,
    0);
```

will send the agent to an agency at “machine.ucdavis.edu” listening on port 5031.

# LẬP TRÌNH MOBILE



Figure 3.1: Architecture of the Mobile-C library.

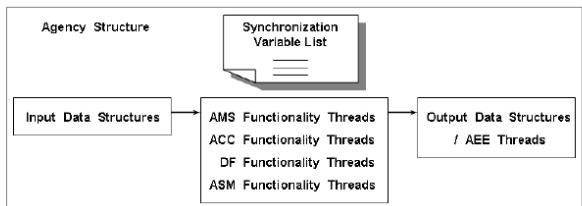
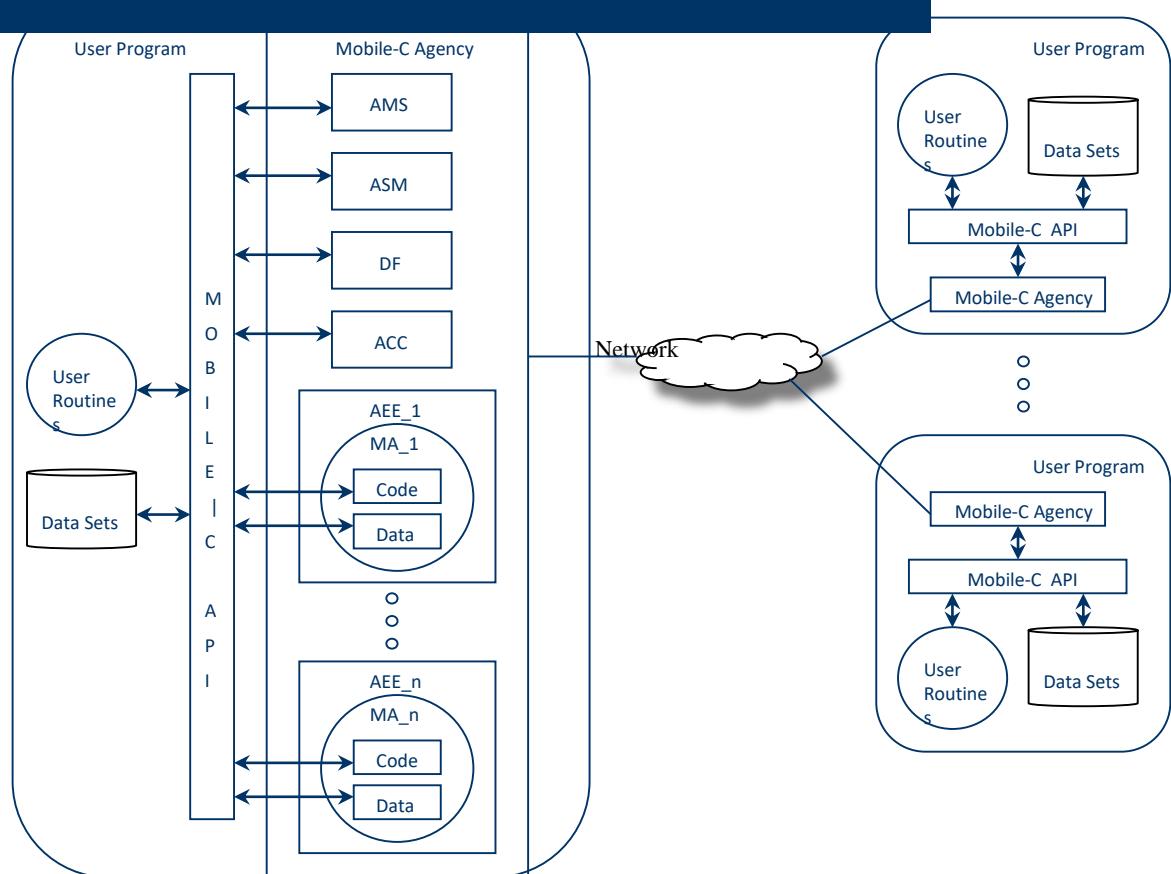
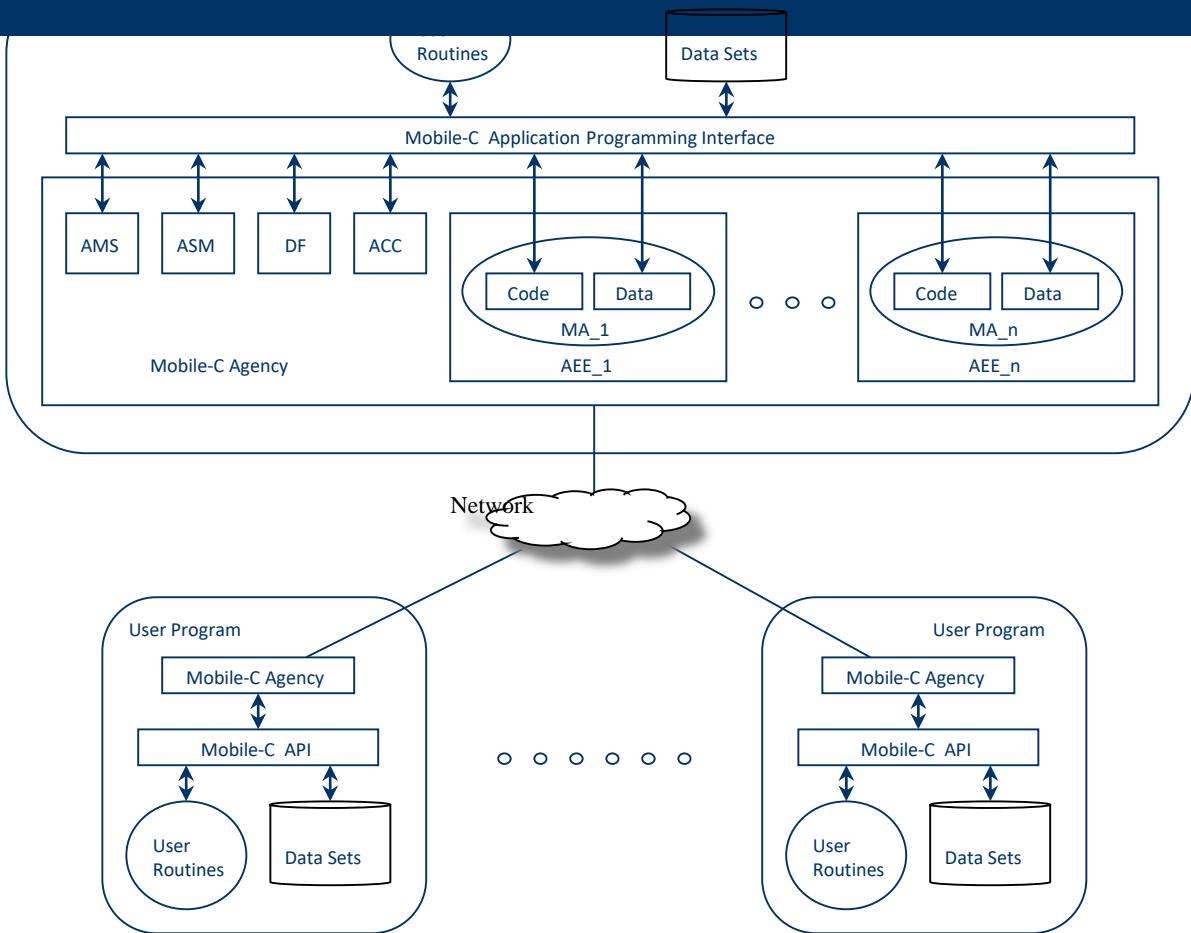


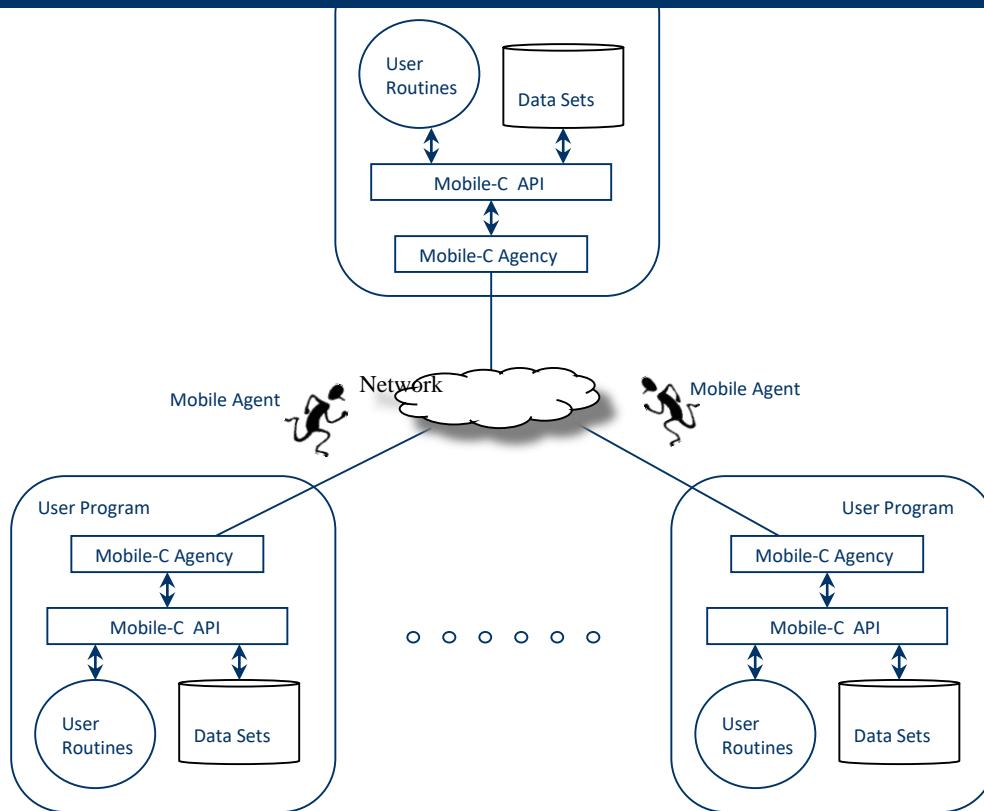
Figure 3.2: Implementation overview of the Mobile-C library.



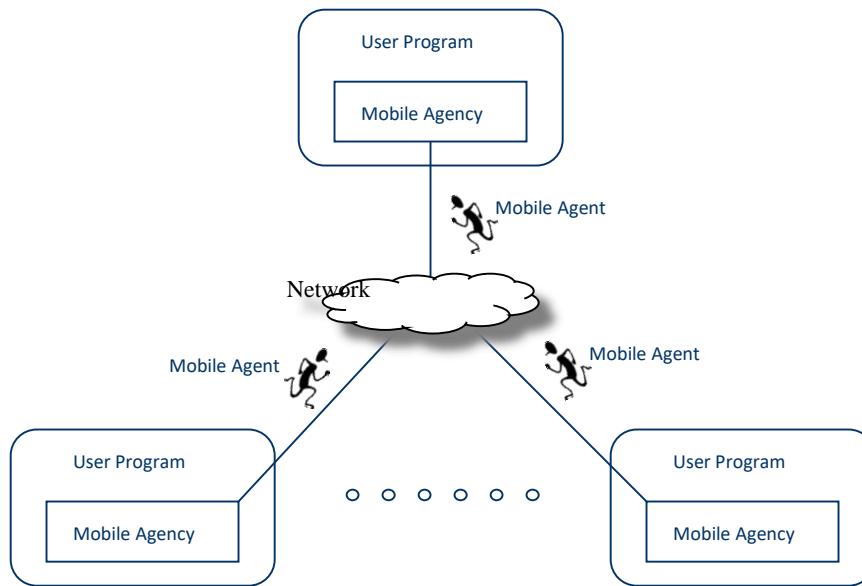
# LẬP TRÌNH MOBILE



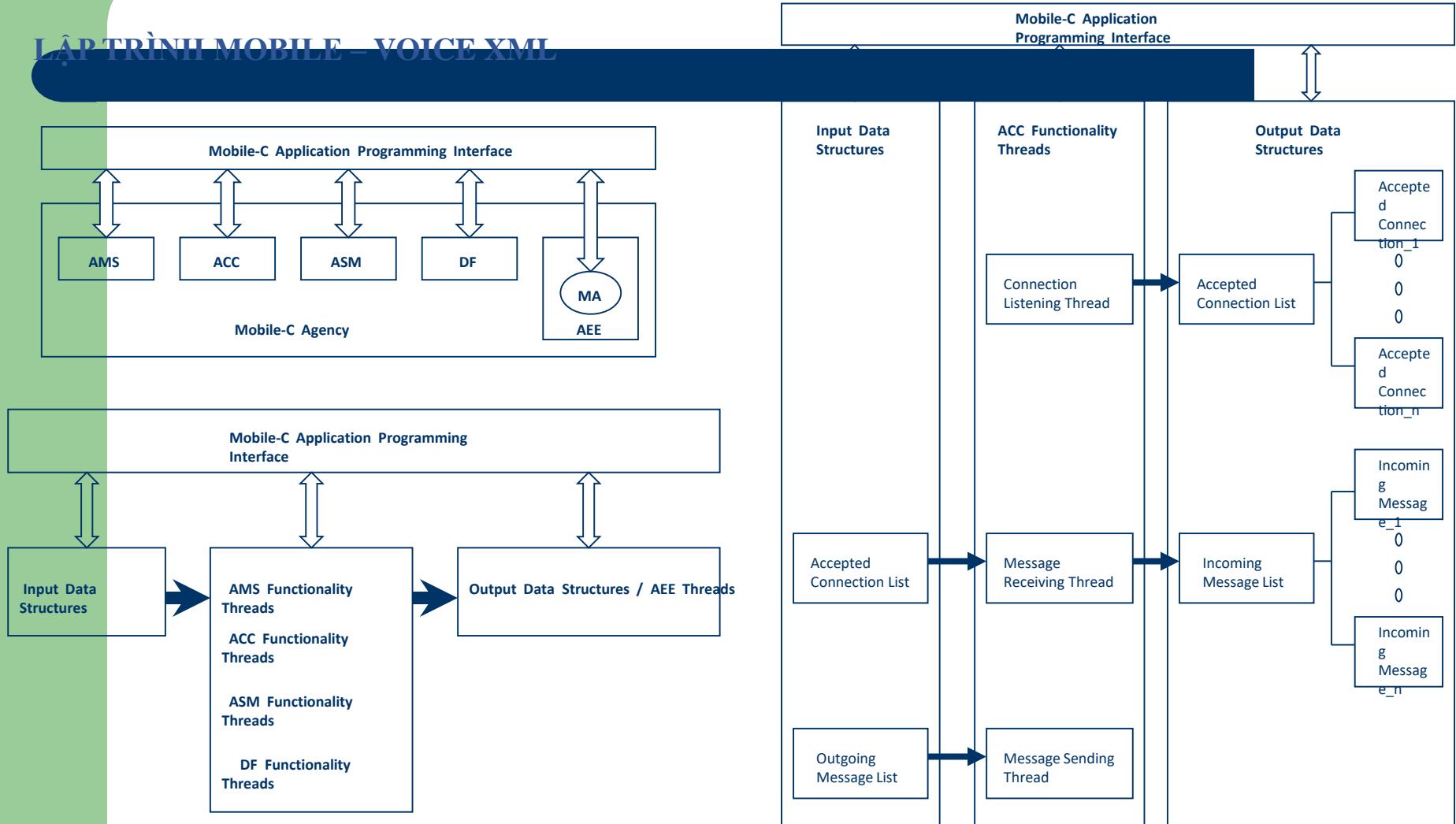
# LẬP TRÌNH MOBILE



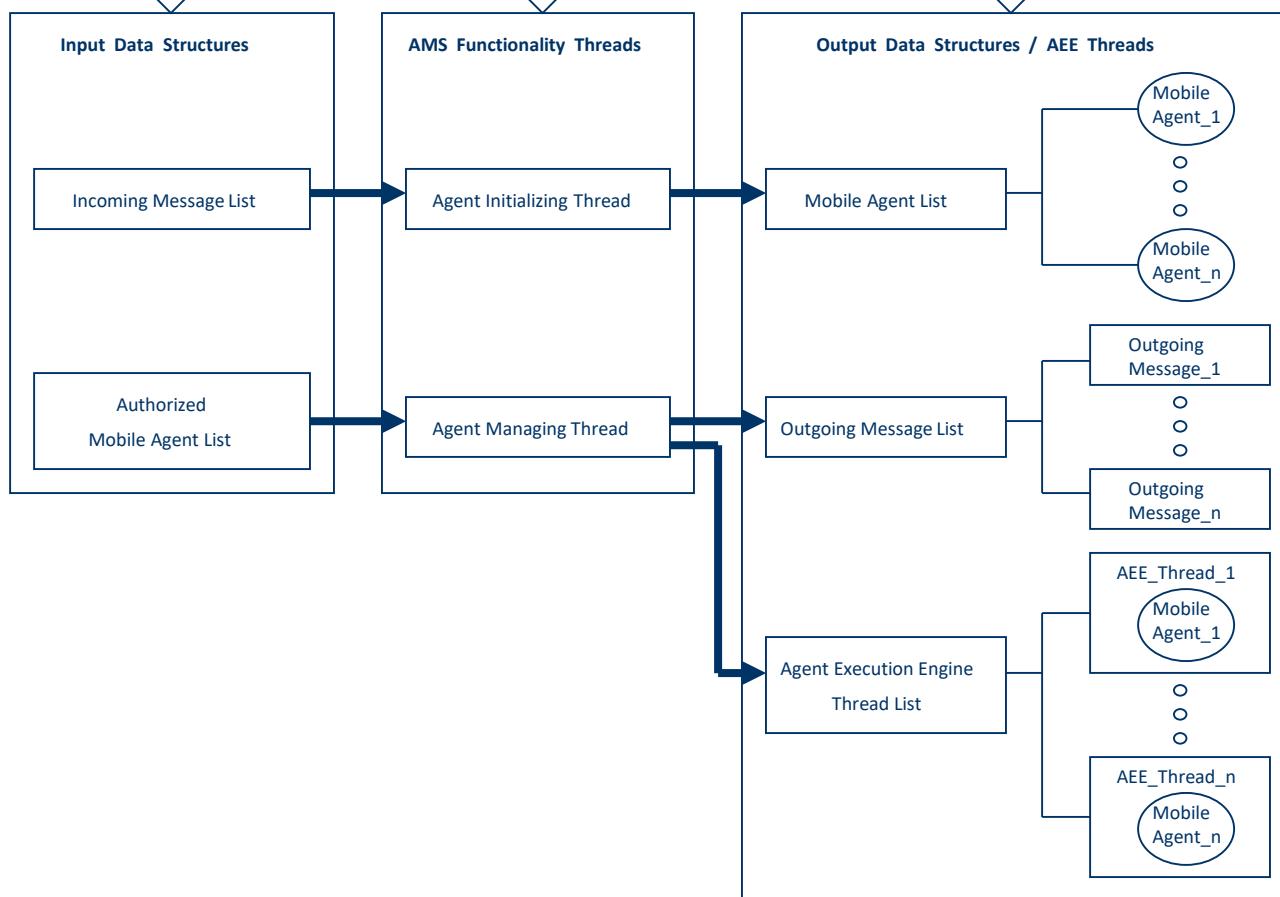
## LẬP TRÌNH MOBILE – VOICE XML



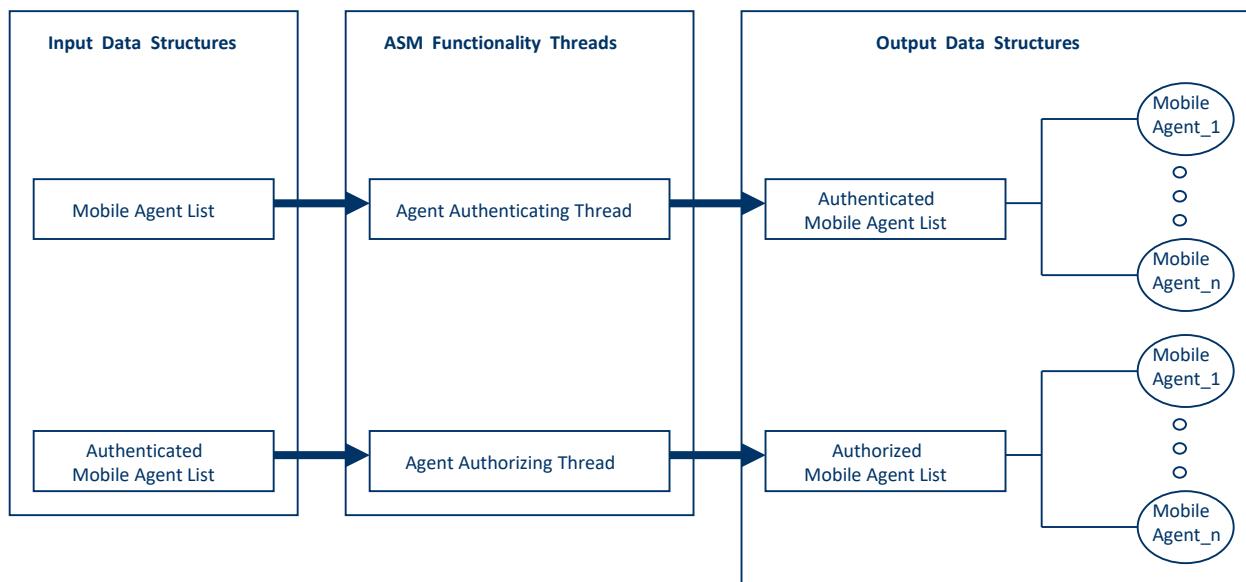
# LẬP TRÌNH MOBILE – VOICE XML



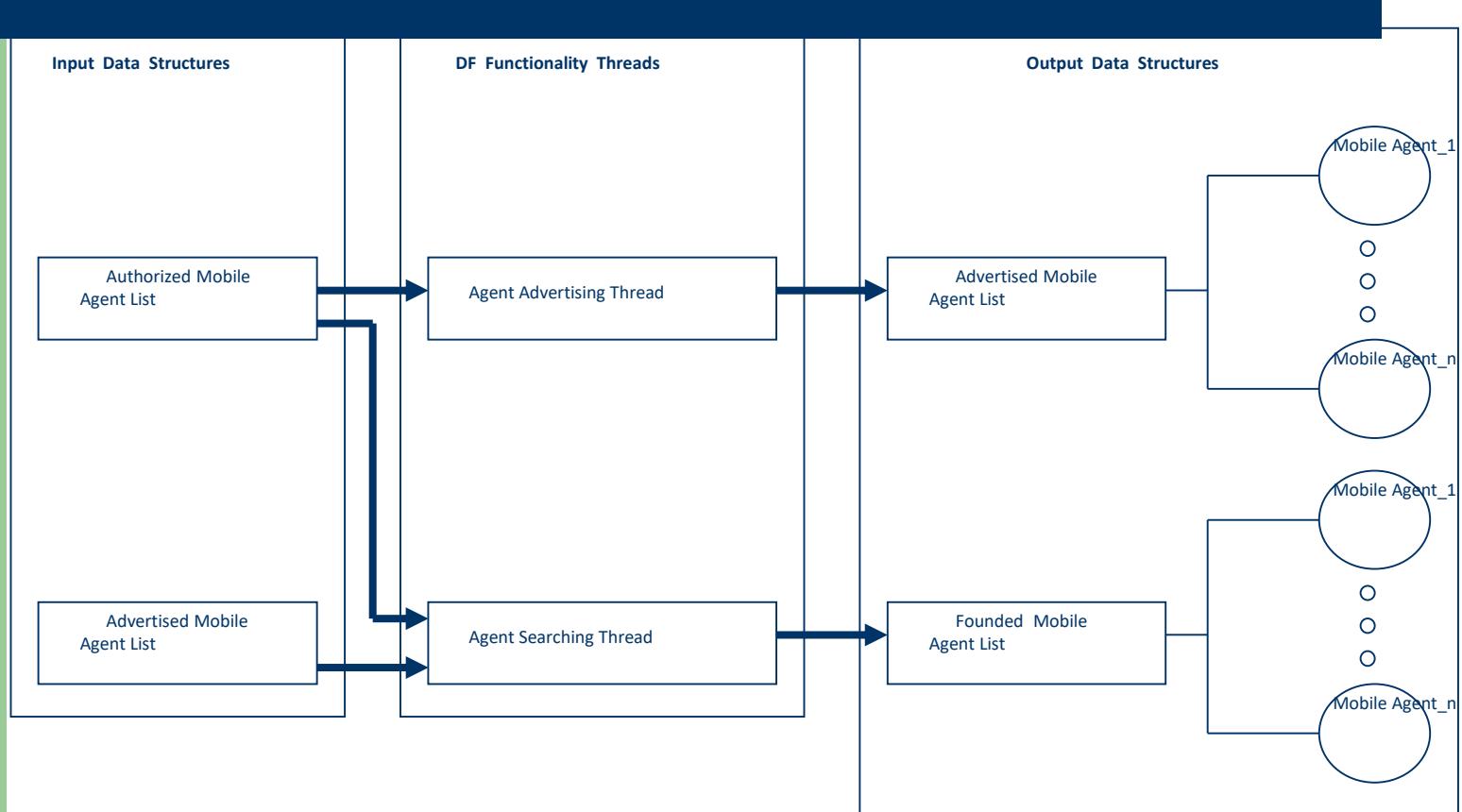
# LẬP TRÌNH MOBILE – VOICE XML



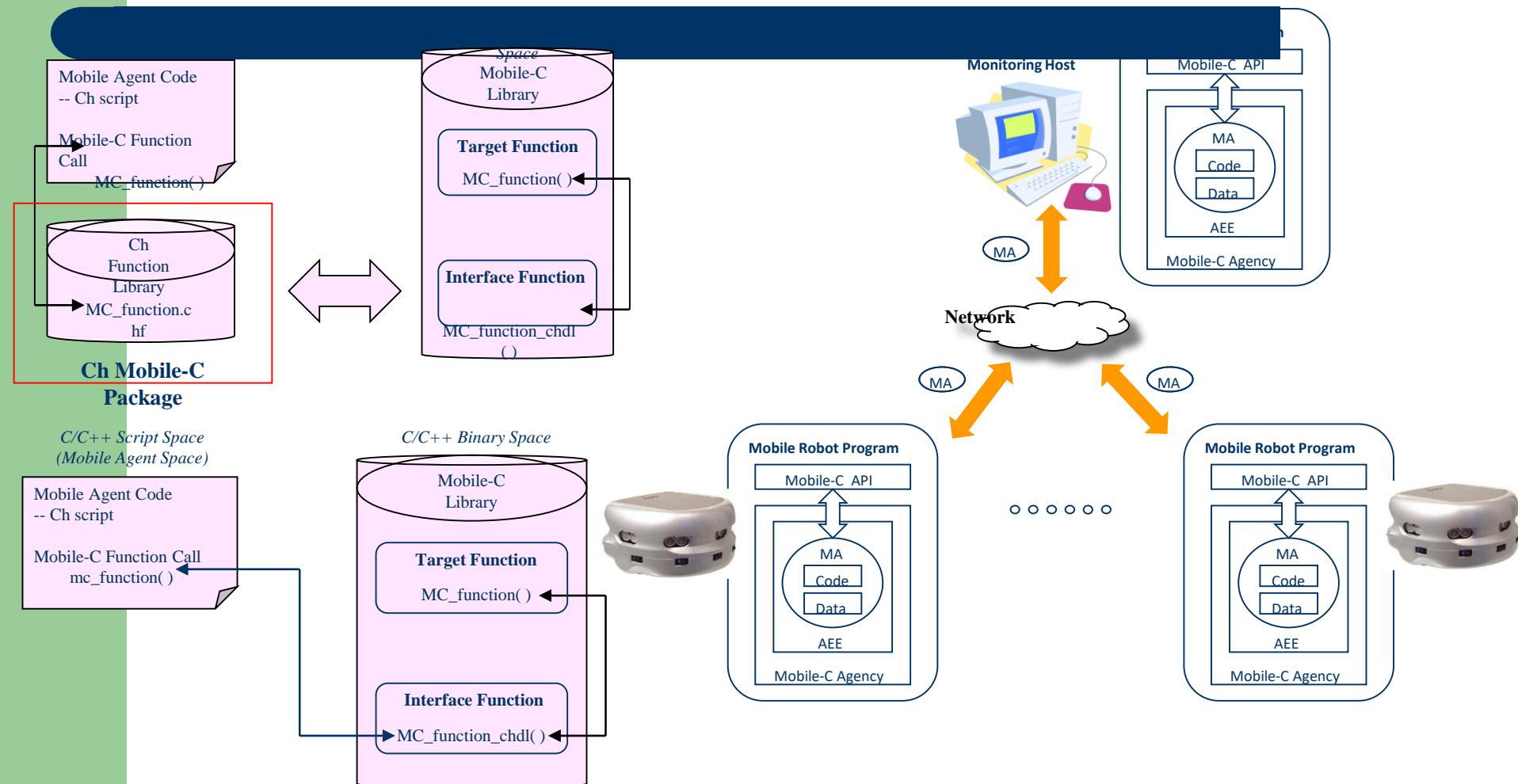
# LẬP TRÌNH MOBILE – VOICE XML



# LẬP TRÌNH MOBILE – VOICE XML



# LẬP TRÌNH MOBILE – VOICE XML



# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

```
MC_SyncInit(500);  
MC_MutexLock(500);
```

## CLIENT Program

```
#include <stdio.h>  
#include <libmc.h>  
- #ifdef _WIN32  
#include <windows.h>  
#else  
#include <unistd.h>  
#endif  
#define WAIT_TIME 5  
int main()  
{  
    MCAgency_t agency;  
    MCAgencyOptions_t options;  
    int local_port=5050;  
    MC_InitializeAgencyOptions(&options);  
    MC_SetThreadOff(&options, MC_THREAD_CP); /* Turn off command prompt */  
    agency = MC_Initialize(local_port, &options);  
  
    printf("Sending sleep agent...\n");  
    /* Note: The third argument of the following function may also be a  
       valid IP address in the form of a string. i.e. 192.168.0.1 */  
    MC_SendAgentFile( agency, "sleep.xml");  
    printf("Sleeping for %d seconds.\n", WAIT_TIME);  
- #ifndef _WIN32  
    sleep(WAIT_TIME);  
#else  
    Sleep(WAIT_TIME * 1000);  
#endif  
    printf("Sending wake-up agent...\n");  
    /* Note: The third argument of the following function may also be a  
       valid IP address in the form of a string. i.e. 192.168.0.1 */  
    MC_SendAgentFile( agency, "wake.xml");  
    printf("Terminating...\n");  
    MC_End(agency);  
    return 0;  
}
```

```
<?xml version="1.0"?>  
  
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">  
  
- <MOBILEC_MESSAGE>  
- <MESSAGE message="MOBILE_AGENT">  
- <MOBILE_AGENT>  
- <AGENT_DATA>  
    <NAME>sleep_agent</NAME>  
    <OWNER>IEL</OWNER>  
    <HOME>localhost:5050</HOME>  
- <TASKS task="1" num="0">  
- <TASK num="0" complete="0" server="localhost:5051">  
    </TASK>  
- <AGENT_CODE>  
- <![CDATA[  
#include <stdio.h>  
  
#define SYNC_ID 55  
int main()  
{  
    int cond_id;  
    printf("Sleep agent has arrived.\n");  
    cond_id = mc_SyncInit(SYNC_ID);  
    if (cond_id != SYNC_ID) {  
        printf("Possible error. Aborting...\n");  
        exit(1);  
    }  
    printf("This is the sleep agent.\n");  
    printf("I am going to sleep now...\n");  
    mc_CondWait(cond_id);  
    printf("This is the sleep agent: I am awake now. Continuing...\n");  
    mc_SyncDelete(cond_id);  
    return 0;  
}  
    ]]>  
- <AGENT_CODE>  
- </TASKS>  
- <AGENT_DATA>  
- <MOBILE_AGENT>  
- <MESSAGE>  
- </MOBILEC_MESSAGE>
```

# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

## SERVER Program

```
#include <libmc.h>
#include <stdio.h>
int main()
{
    MCAgency_t agency;
    MCAgencyOptions_t options;
    int i;
    int local_port=5051;

    MC_InitializeAgencyOptions(&options);

    for (i = 0; i < MC_THREAD_ALL; i++) {
        MC_SetThreadOn(&options, i);
    }
    /* if the following line is uncommented, the command prompt
     * will be disabled. */
/*    MC_SetThreadOff(&options, MC_THREAD_CP); */

    agency = MC_Initialize(
        local_port,
        &options);

    /* Wait forever... */
    MC_MainLoop(agency);
    return 0;
}
```

```
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">
<MOBILEC_MESSAGE>
- <MESSAGE message="MOBILE_AGENT">
-   <MOBILE_AGENT>
-     <AGENT_DATA>
-       <NAME>wake agent</NAME>
-       <OWNER>IEL</OWNER>
-       <HOME>localhost:5050</HOME>
-       <TASKS task="1" num="0">
-         <TASK num="0" complete="0" server="localhost:5051">
-           </TASK>
-         <AGENT_CODE>
-           <![CDATA[
#include <stdio.h>
#define SYNC_ID 55
int main()
{
    int cond_id;
    int err;
    cond_id = SYNC_ID;
    printf("This is the wake agent.\n");
    err = mc_CondSignal(cond_id);
    if(err) {
        printf("Error signalling condition variable!\n");
    }
    return 0;
}
]]>
</AGENT_CODE>
</TASKS>
</AGENT_DATA>
</MOBILE_AGENT>
</MESSAGE>
</MOBILEC_MESSAGE>
```

# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

```
#include <stdio.h>
#include <libmc.h>
#ifndef _WIN32
#include <windows.h>
#endif
#define WAIT_TIME 2
int main()
{
    MCAgency_t agency;
    MCAgencyOptions_t options;
    int local_port=5050;

    MC_InitializeAgencyOptions(&options);
    MC_SetThreadOff(&options, MC_THREAD_CP); /* Turn off command prompt */
    agency = MC_Initialize(local_port, &options);

    printf("Sending sleep agent...\n");
    /* Note: The third argument of the following function may also be a
       valid IP address in the form of a string. i.e. 192.168.0.1 */
    MC_SendAgentFile( agency, "sleep.xml");
    printf("Sleeping for %d seconds.\n", WAIT_TIME);
#ifndef _WIN32
    sleep(WAIT_TIME);
#else
    Sleep(WAIT_TIME * 1000);
#endif
    printf("Sending wake-up agent...\n");
    /* Note: The third argument of the following function may also be a
       valid IP address in the form of a string. i.e. 192.168.0.1 */
    MC_SendAgentFile( agency, "wake.xml");
    MC_End(agency);
    return 0;
}
```

CLIENT PROGRAM

```
<?xml version="1.0"?>
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">
<MOBILEC_MESSAGE>
  <MESSAGE message="MOBILE_AGENT">
    <MOBILE_AGENT>
      <AGENT_DATA>
        <NAME>sleep_agent</NAME>
        <OWNER>IEL</OWNER>
        <HOME>localhost:5050</HOME>
        <TASKS task="1" num="0">
          <TASK num="0" complete="0" server="localhost:5051">
            </TASK>
        <AGENT_CODE>
          <![CDATA[
#include <stdio.h>
int main()
{
    int mutex_id;
    printf("Sleep agent has arrived.\n");
    mutex_id = mc_SyncInit(55);
    if (mutex_id != 55) {
        printf("Possible error. Aborting...\n");
        exit(1);
    }
    printf("This is agent 1.\n");
    printf("Agent 1: I am locking the mutex now.\n");
    mc_MutexLock(mutex_id);
    printf("Agent 1: Mutex locked. Perform protected operations here\n");
    printf("Agent 1: Waiting for 5 seconds...\n");
    sleep(5);
    printf("Agent 1: Unlocking mutex now...\n");
    mc_MutexUnlock(mutex_id);

    return 0;
}
]]>
        </AGENT_CODE>
      </TASKS>
    <AGENT_DATA>
    </MOBILE_AGENT>
  </MESSAGE>
</MOBILEC_MESSAGE>
```

sleep.xml

# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

```
- /* mc_sample_app.c
 *
 * This sample program uses the Mobile C library to build
 * a simple command-line driven client/server app.
 *
 * 12/15/2006
 * */
 *
#include <libmc.h>
#include <stdio.h>
int main()
{
    MCAgency_t agency;
    MCAgencyOptions_t options;
    int i;
    int local_port=5051;
    MC_InitializeAgencyOptions(&options);

    for (i = 0; i < MC_THREAD_ALL; i++) {
        MC_SetThreadOn(&options, i);
    }
    /* If the following line is uncommented, the command prompt
     * will be disabled. */
    /*MC_SetThreadOff(&options, MC_THREAD_CP); */

    agency = MC_Initialize(
        local_port,
        &options);

    /* Wait forever... */
    MC_MainLoop(agency);
    return 0;
}
```

SERVER PROGRAM

```
<?xml version="1.0"?>
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">

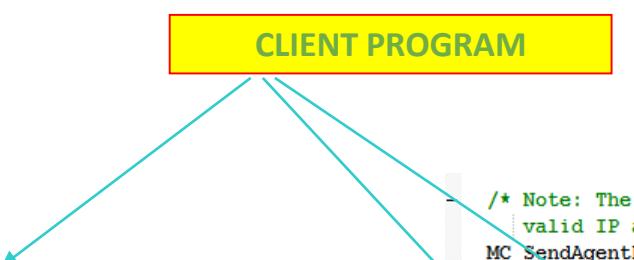
<MOBILEC_MESSAGE>
    <MESSAGE message="MOBILE_AGENT">
        <MOBILE_AGENT>
            <AGENT_DATA>
                <NAME>wake_agent</NAME>
                <OWNER>IEL</OWNER>
                <HOME>localhost:5050</HOME>
                <TASKS task="1" num="0">
                    <TASK num="0" complete="0" server="localhost:5051">
                </TASK>
                <AGENT_CODE>
                    <![CDATA[
#include <stdio.h>
int main()
{
    int mutex_id;
    mutex_id = 55;
    printf("Agent 2: Has arrived");
    printf("Agent 2: Attempting to lock the mutex...\n");
    mc_MutexLock(mutex_id);
    printf("Agent 2: Mutex locked.\n");
    printf("Agent 2: Perform protected operations here.\n");
    sleep(5);
    mc_MutexUnlock(mutex_id);
    printf("Agent 2: Mutex Unlocked\n");
    mc_SyncDelete(mutex_id);

    return 0;
}
                ]]>
            </AGENT_CODE>
        </TASKS>
        <AGENT_DATA>
        </MOBILE_AGENT>
    </MESSAGE>
</MOBILEC_MESSAGE>
```

wake.xml

# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

```
*  
* This sample program uses the Mobile C library to build  
* a simple command-line driven client/server app.  
*  
* 12/15/2006  
* */  
  
- #ifndef _WIN32  
#include <libmc.h>  
#else  
#include <libmc.h>  
#include <windows.h>  
#endif  
#include <stdio.h>  
- #ifndef _WIN32  
#include <unistd.h>  
#endif  
  
int main()  
{  
    /* Init the agency */  
    MCAgency_t agency;  
    MCAgencyOptions_t options;  
    int local_port=5051;  
  
    MC_InitializeAgencyOptions(&options);  
    MC_SetThreadOff(&options, MC_THREAD_CP); /* Turn off command prompt */  
    agency = MC_Initialize(local_port, &options);  
  
    /* Note: The third argument of the following function may also be a  
     * valid IP address in the form of a string. i.e. 192.168.0.1 */  
    MC_SendAgentFile( agency, "comm_agent.xml");  
- #ifndef _WIN32  
    sleep(1);  
#else  
    Sleep(1000);  
#endif  
    /* Note: The third argument of the following function may also be a  
     * valid IP address in the form of a string. i.e. 192.168.0.1 */  
    MC_SendAgentFile( agency, "agent_1.xml");  
- #ifndef _WIN32  
    sleep(1);  
#else  
    Sleep(1000);  
#endif  
    /* Note: The third argument of the following function may also be a  
     * valid IP address in the form of a string. i.e. 192.168.0.1 */  
    MC_SendAgentFile( agency, "agent_2.xml");  
  
    MC_End(agency);  
    return 0;  
}
```



# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

```
#else
#include <libmc.h>
#endif
#include <embedch.h>
#include <stdio.h>
#ifndef _WIN32
#include <unistd.h>
#endif

int main()
{
    MCAgency_t agency;
    MCAgencyOptions_t options;
    int i;
    int local_port=5050;
    /* We want all the threads on: EXCEPT, the command prompt thread */
    MC_InitializeAgencyOptions(&options);
    for (i = 0; i < MC_THREAD_ALL; i++) {
        MC_SetThreadOn(&options, i);
    }
    /* If the following line is uncommented, the command prompt will
     * be turned off */
    /*MC_SetThreadOff(&options, MC_THREAD_CP); */

    /* Init the agency */

    agency = MC_Initialize(
        local_port,
        &options);

    MC_MainLoop(agency);
    return 0;
}
```

SERVER PROGRAM

```
<?xml version="1.0"?>
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">
<MOBILEC_MESSAGE>
    <MESSAGE message="MOBILE_AGENT">
        <MOBILE_AGENT>
            <AGENT_DATA>
                <NAME>CommAgent1</NAME>
                <OWNER>IEL</OWNER>
                <HOME>localhost:5051</HOME>
                <TASKS task="1" num="0">
                    <TASK num="0" complete="0" server="localhost:5050">
                    </TASK>
                <AGENT_CODE>
                    <![CDATA[
#include <stdio.h>
#define MC_BARRIER_ID 56
#define NUM_PROCS 2
int main()
{
    printf("Comm agent: Setting up the MC_Barriers now...\n");
    mc_BarrierInit(MC_BARRIER_ID, NUM_PROCS);
    mc_BarrierInit(MC_BARRIER_ID + 1, NUM_PROCS);
    mc_BarrierInit(MC_BARRIER_ID + 2, NUM_PROCS);
    printf("MC_Barrier Initialized.\n");
    return 0;
}
                    ]]>
                </AGENT_CODE>
            </TASKS>
            <AGENT_DATA>
        </MOBILE_AGENT>
    </MESSAGE>
</MOBILEC_MESSAGE>
```

comm\_agent.xml

# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

```
<?xml version="1.0"?>
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">

<MOBILEC_MESSAGE>
  <MESSAGE message="MOBILE_AGENT">
    <MOBILE_AGENT>
      <AGENT_DATA>
        <NAME>agent1</NAME>
        <OWNER>IEL</OWNER>
        <HOME>localhost:5051</HOME>
        <TASKS task="1" num="0">
          <TASK num="0" complete="0" server="localhost:5050" persistent="1">
            </TASK>
        <AGENT_CODE>
          <![CDATA[
#include <stdio.h>
#define MC_BARRIER_ID 56

int main()
{
    printf("Agent 1: Approaching Barrier 1...\n");
    mc_Barrier(MC_BARRIER_ID);
    printf("Agent 1: Past Barrier 1.\n");
    sleep( 10 );
    printf("Agent 1: At Barrier 2.\n");
    mc_Barrier(MC_BARRIER_ID + 1);
    printf("Agent 1: Past Barrier 2.\n");
    sleep( 10 );
    printf("Agent 1: At Barrier 3.\n");
    mc_Barrier(MC_BARRIER_ID + 2);
    printf("Agent 1: Past Barrier 3.\n");
    return 0;
}
          ]]>
      </AGENT_CODE>
    </TASKS>
  </AGENT_DATA>
</MOBILE_AGENT>
</MESSAGE>
</MOBILEC_MESSAGE>
```

agent1.xml

```
<?xml version="1.0"?>
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">

<MOBILEC_MESSAGE>
  <MESSAGE message="MOBILE_AGENT">
    <MOBILE_AGENT>
      <AGENT_DATA>
        <NAME>agent2</NAME>
        <OWNER>IEL</OWNER>
        <HOME>localhost:5051</HOME>
        <TASKS task="1" num="0">
          <TASK num="0" complete="0" server="localhost:5050" persistent="1">
            </TASK>
        <AGENT_CODE>
          <![CDATA[
#include <stdio.h>
#define MC_BARRIER_ID 56

int main()
{
    printf("Agent 2: Approaching Barrier 1...\n");
    mc_Barrier(MC_BARRIER_ID);
    printf("Agent 2: Past Barrier 1.\n");
    sleep( 1 );
    printf("Agent 2: At Barrier 2.\n");
    mc_Barrier(MC_BARRIER_ID + 1);
    printf("Agent 2: Past Barrier 2.\n");
    sleep( 1 );
    printf("Agent 2: At Barrier 3.\n");
    mc_Barrier(MC_BARRIER_ID + 2);
    printf("Agent 2: Past Barrier 3.\n");
    return 0;
}
          ]]>
      </AGENT_CODE>
    </TASKS>
  </AGENT_DATA>
</MOBILE_AGENT>
</MESSAGE>
</MOBILEC_MESSAGE>
```

agent2.xml

# LẬP TRÌNH MOBILE – SYNCHRONIZATION SUPPORT IN THE MOBILE-C LIBRARY

```
#else
#include <libmc.h>
#endif
#include <embedch.h>
#include <stdio.h>
#ifndef _WIN32
#include <unistd.h>
#endif

int main()
{
    MCAgency_t agency;
    MCAgencyOptions_t options;
    int i;
    int local_port=5050;
    /* We want _all_ the threads on: EXCEPT, the command prompt thread */
    MC_InitializeAgencyOptions(&options);
    for (i = 0; i < MC_THREAD_ALL; i++) {
        MC_SetThreadOn(&options, i);
    }
    /* If the following line is uncommented, the command prompt will
     * be turned off */
    /*MC_SetThreadOff(&options, MC_THREAD_CP); */

    /* Init the agency */

    agency = MC_Initialize(
        local_port,
        &options);

    MC_MainLoop(agency);
    return 0;
}
```

## SERVER PROGRAM

# LẬP TRÌNH MOBILE – SECURITY MODULE

## SERVER PROGRAM

## CLIENT PROGRAM

```
#include <stdio.h>
#include <stdlib.h>
#include <libmc.h>
#include <string.h>

int main()
{
    MCAgency_t agency;
    MCAgencyOptions_t options;
    int local_port=5125;
    int remote_port = 5126;
    char remote_host[] = "localhost";

    MC_InitializeAgencyOptions(&options);
    strcpy( options.passphrase, "alpha1234");

    MC_SetThreadOff(&options, MC_THREAD_CPT); /* Turn off command prompt */
    agency = MC_Initialize(local_port, &options);
    /* Note: The third argument of the following function may also be a
       valid IP address in the form of a string. i.e. 192.168.0.1 */
    printf("Sending agent...");
    MC_SendAgentFile(agency, "test1.xml");
    printf(" Done.\n");
    MC_End(agency);
    exit(0);
}
```

```
#include <stdio.h>
#include <libmc.h>
#ifndef _WIN32
#include <windows.h>
#endif

int main()
{
    MCAgency_t agency;
    int local_port = 5126;
    //unsigned char passphrase[] = "alpha1234";
    MCAgencyOptions_t options;

    MC_InitializeAgencyOptions(&options);
    strcpy(options.passphrase, "alpha1234");

    agency = MC_Initialize(local_port, &options);

    MC_MainLoop(agency);

    MC_End(agency);
    return 0;
}
```

# LẬP TRÌNH MOBILE – SECURITY MODULE

```
<!DOCTYPE myMessage SYSTEM "mobilec.dtd">

- <MOBILEC_MESSAGE>
- <MESSAGE message="MOBILE_AGENT">
- <MOBILE_AGENT>
-   <AGENT_DATA>
-     <NAME>mobagent1</NAME>
-     <OWNER>IEL</OWNER>
-     <HOME>localhost:5050</HOME>
-     <TASKS task="1" num="0">
-       <TASK num="0" return="no-return" complete="0" server="localhost:5126" />
-     <AGENT_CODE>
-       <![CDATA[
#include <stdio.h>
#include <math.h>
int main()
{
    char* str;
    printf("Hello World!\n");
    printf("This is mobagent1 from the agency at port 5125.\n");
    printf("I am performing the task on the agency at port 5126 now.\n");
    printf("%f\n", hypot(1,2));

    return 0;
}
]]>
</AGENT_CODE>
</TASKS>
</AGENT_DATA>
</MOBILE_AGENT>
</MESSAGE>
</MOBILEC_MESSAGE>
```

Test1.xml

```
<?xml version="1.0"?>

<!DOCTYPE myMessage SYSTEM "mobilec.dtd">

- <MOBILEC_MESSAGE>
- <MESSAGE message="MOBILE_AGENT">
-   <MOBILE_AGENT>
-     <AGENT_DATA>
-       <NAME>mobagent1</NAME>
-       <OWNER>IEL</OWNER>
-       <HOME>localhost:5050</HOME>
-       <TASK task="1" num="0">
-         <DATA dim="0" name="no-return" complete="0" server="localhost:5051">
-           </DATA>
-         <AGENT_CODE>
-           <![CDATA[
#include <stdio.h>

struct mytime{
    unsigned long sec;
    long usec;
};

int main()
{
    struct mytime t;
    mc_gettimeofday(&t);
    printf("The time is %d, %d", t.sec, t.usec);
    printf("Hello World!\n");
    printf("This is mobagent1 from the agency at port 5050.\n");
    printf("I am performing the task on the agency at port 5051 now.\n");
    mc_gettimeofday(&t);

}
]]>
</AGENT_CODE>
</TASK>
</AGENT_DATA>
</MOBILE_AGENT>
</MESSAGE>
</MOBILEC_MESSAGE>
```

Test\_gettimeofday.xml

# LẬP TRÌNH MOBILE – VOICEXML MODULE

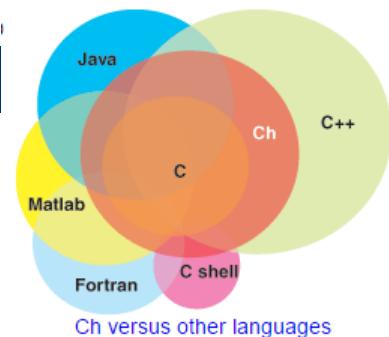
# LẬP TRÌNH MOBILE – VOICE XML

Hệ thống khóa đa mức

Hệ thống bán lẻ thanh toán qua điện thoại di động

Máy dịch ngôn ngữ tự động

Điều khiển các thiết bị điện trong gia đình thông qua công nghệ voicexml



## SoftIntegration Products

### Base Products

Ch Standard 7.5.2 (Free)

Ch Professional 7.5.2

Ch Student 7.5.2 (Free)

Ch SDK 7.5.2 (Free)

Ch Edition Comparison

### Embedded Ch

Embedded Ch 7.5.2

### Toolkits/Packages

Ch Mechanism 3.0

Ch Control System 2.5

Ch NAG Statistics 1.1

ChExcel 1.2 (Free)

Ch CGI 3.7 (Free)

### C++ Library

C++ Graphical Library 3.0

### Services

Web services and Web-based computing

### Others

Third Party Solutions and User Contributed Code

Product Awards

## Third Party Solutions and User Contributed Code

- [Arduino](#)
- [Excel Spreadsheet](#)
- [Internet](#)
- [Mechanism Design and Analysis](#)
- [Database](#)
- [Numerical Analysis](#)
- [Simulation](#)
- [2D/3D Plotting, Graphics, Computer Vision, and Image Processing](#)
- [Mobile Computing](#)
- [Multimedia](#)
- [Graphical User Interface \(GUI\)](#)
- [XML](#)
- [Real-Time Computing](#)
- [3D Audio](#)
- [3D Audio](#)
- [Motion Control](#)
- [Digital Signal Processing](#)
- [Data Acquisition](#)
- [Robotics](#)
- [Java](#)
- [MPI \(Message Passing Interface\)](#)
- [Integrated Development Environment \(IDE\)](#)
- [Compression and Decompression](#)
- [Character Encoding Conversion](#)
- [Regular Expressions](#)
- [Record-Jar Structured Text File Format](#)
- [Dynamics](#)
- [SWIG Ch binding](#)

# LẬP TRÌNH C

## Third Party Solutions and User Contributed Code

### Arduino

[Ch Arduino -- included in C-STEM studio](#)

### Excel Spreadsheet

[ChExcel to Manipulate Microsoft Excel Using C/C++ Scripts](#)

### Internet

[Mobile-C\\_ a multi-agent platform for mobile C/C++ agents](#)

[Ch CGI \(Common Gateway Interface\) to Web servers](#)

[Ch cURL package for FTP, FTPS, HTTP, HTTPS, GOPHER, TELNET, DICT, FILE and LDAP processing.](#)

### Mechanism Design and Analysis

[Whitworth Quick Return Mechanism](#)

[Kinematic Synthesis of Mechanisms](#)

[Computer-Aided Mechanism Design and Analysis Using ChExcel](#)

### Numerical Analysis

[Ch LAPACK for Numerical Analysis](#) (Bundled with Ch Professional Edition)

[Ch GMP for Multiple Precision Arithmetic](#)

[Ch GSL for GNU Scientific Library](#)

[Ch GAUL \(Genetic Algorithm Utility Library\)](#)

[ChExcel to Manipulate Microsoft Excel Using C/C++ Scripts](#)

### Simulation

[RecurDyn/CoLink -- a complete CAE solution for design and simulation of multi-body dynamic \(MBD\) and control systems](#)

### 2D/3D Plotting, Graphics, Computer Vision, and Image Processing

[Ch Plotting](#) (Bundled with Ch Professional Edition)

[Boxplot for boxes and whisker points for vector data](#)

[Ch Plotting using GTK+](#)

[Ch OpenGL](#) (Bundled with Ch) and [Nate Robins' OpenGL Tutorial Demo Programs](#) (Tested in Ch)

[Ch imagemagick](#)

[Ch Windows](#)

[Ch OpenCV](#) (Intel Open Source Computer Vision for Image Processing and Computer Vision)

[Ch PNG](#)

[Ch TIFF](#)

[Ch JPEG](#)

[Ch Ming for generating SWF \("Flash"\) format movies](#)

[DISLIN high-level plotting library](#)

### Mobile Computing

[Mobile-C \(Mobile Agent-Based Computing with Mobile C/C++ Code\)](#)

### Multimedia

[Ch SDL \(Simple Directmedia Layer\)](#)

### Graphical User Interface (GUI)

[Ch GTK+](#)

[Ch X/Motif \(Bundled with Ch\)](#)

[Ch Win32 \(Bundled with Ch\)](#)

### XML

[Ch XML Package for Oracle C/C++ XDK](#)

[Ch Libxml2 Package for XML C parser for the Gnome project](#)

[Ch Mini-XML Package](#)

### Real-Time Computing

[Ch RTAI \(Realtime Application Interface\) for Linux](#)

RTLinux API is supported readily in Ch.

### 3D Audio

[Ch OpenAL \(3D Audio API\)](#)

### Motion Control

[PSIM powered with Embedded Ch from Powersim Inc.](#)

[Ch NI-Motion \(National Instruments NI-Motion FlexMotion\)](#)

[Ch PMAC PCOM \(Delta Tau Data Systems's PMAC PCOM\)](#)

### Digital Signal Processing

[SigLib DSP library supports Ch](#)

### Data Acquisition

[Velleman Ch K8055 for USB Experiment Interface Board K8055](#)

[Ch NI-DAQ \(National Instruments NI-DAQ\)](#)

[Ch LabVIEW \(Embed Ch into LabVIEW\)](#)

### Robotics

[Ch Robot Controller for Lego Mindstorms NXT and EV3](#)

[Ch Mindstorms Package for NXT and EV3 -- included in C-STEM Studio](#)

[Ch Linkbot Controller for Linkbot](#)

[Ch Finch robot](#)

[Ch RoboTalk to communicate with RoboWorks models in real-time.](#)

### Java

[Ch Java](#)

### MPI

[Ch MPI \(Message Passing Interface\)](#)

### Integrated Development Environment (IDE)

Edit and run Ch programs, with syntax highlighting for [keywords](#) and shebang "#!/bin/ch", and the output displayed within a Graphical User Interface (GUI).

[ChIDE](#), it is now integrated into Ch Professional and Student Editions

[ChScite IDE for Ch with the user interface in the local language \(Windows binary and source code for other platforms are available\)](#) (free)

[Zeus Editor for Windows](#) (Download [zeus 3.92 or above](#) supporting Ch)

[Crimson Editor for Windows](#) with [instructions to use Ch](#) and a [tutorial on how to get started](#) (free)

[EditPad Pro for Windows and Linux](#)

[Code Forge for Linux and Unix](#)

[EmEditor for Windows](#)

[EditPlus for Windows](#) with [Ch syntax files and configuration instructions](#)

[SlickEdit for Windows, Unix and Mac OS X](#)

[UltraEdit for Windows](#)

C-Free for Windows, providing both [English Edition](#) and [Chinese Edition](#)

[NEdit](#) for Unix and Mac OS X with [Ch Syntax file](#) (free)

[ConTEXT Editor for Windows](#) with [instructions to use Ch](#) (free)

[Magic C++ for Windows](#)(with cygwin)/Linux/Unix. It also supports [Chinese language](#).

[Kode for Windows and Linux](#)

[TextWrangler for Mac OS X](#) (free)

### Compression and Decompression

[Ch zlib](#)

### Character Encoding Conversion

[Ch libiconv](#)

### Regular Expressions

[Ch PCRE \(Perl Compatible Regular Expressions\)](#)

### Record-Jar Structured Text File Format

[Ch Open-RJ for readers of the Record-Jar structured text file format.](#)

### Dynamics

[Using Ch with Autolev to solve dynamics equations.](#)

### SWIG Ch Binding

[Chase project using SWIG to bind Ch with third party libraries.](#)

## Tài liệu tham khảo

- [1]. Bài giảng lập trình C, Nguyễn Đình Thuân, Đại học Nha Trang, 2008.
- [2] Bài giảng Kỹ thuật lập trình, Nguyễn Ngọc Phương, Học viện bưu chính viễn thông.
- [3] Bài giảng toán rời rạc, Đỗ Như An, Đại học Nha Trang, 2008.
- [4] Lập trình C cơ bản. (<http://tuhocanninhmang.com>, <http://www.cplusplus.com>)
- [5] The algorithm Design Manual, Steven S. Skiena, 2008
- [6] Lập trình C mô phỏng thế giới thật.
- [7] Numerical Recipes in C, the Art of Scientific Computing Second Edition